Industry Foundation Classes - Release 2.0

IFC Specification Development Guide

15-Mar-99

International Alliance for Interoperability
Enabling Interoperability in the AEC/FM Industry

Industry Foundation Classes - Release 2.0

IFC Specification Development Guide

Enabling Interoperability in the AEC/FM Industry

Document Editor

Editor	Jeffrey Wix (primary) / Richard See (secondary)
Development committee	Specification Task Force

Document Control

Project reference	IFC Release 2.0
Document reference	IFC Specifications Development Guide
Document version	Final
Release date	15-Mar-99
Status	Final
Distribution	Public
Distribution format	PDF file

Revisions

Rev.	Person	Date	Description
Alpha	Jeffrey Wix	10-Aug-98	Alpha release
Beta, d1	Jeffrey Wix	19-Dec-98	Beta release
Beta, d2	Richard See	10-Jan-99	Std. front end, page layout, fix figures indexing
Final, d3	Jeffrey Wix	26-Feb-99	Various changes
Final, d4	Richard See	15-Mar-99	Final Release

Preface to the 2nd Edition

The first version of the IFC Specification Development Guide[1] was issued in conjunction with IFC Release 1.5 and contained guidance on project development and on the software and modeling technologies used in project development.

For project development, the Guide drew heavily on the experiences that project teams had gained in the projects leading to the IFC Release 1.0 specification. For technology usage, several of the Appendices in the Guide were able to expand upon work carried out previously in connection with the development of the Building Construction Core Model (the proposed ISO 10303 part 106 issued as document WG3/N599 version T300) and supported under the 'Computerized Exchange of Information in Construction' project by the Building Research Establishment in the UK. Experiences in the ATLAS and COMBI projects funded by the European Union were also invaluable in formulating the original Guide.

In this second edition, the guidance on project development has been completely redeveloped and now follows a more clearly defined methodological approach. The process for IFC project development has matured since Release 1.0 and the experiences from work on IFC Release 2.0 projects and IFC Release 3.0 projects are now available. A formal process model for development has been created. An explanation of what is needed at each stage of the process is given together with the expected result. The process model also identifies the performers in the process; that is, which groups are involved. Most processes now also include examples to demonstrate the format of the expected deliverables at each stage. Most of the examples are taken from a single IFC development project and are therefore consistent. It has not yet been possible to draw all examples from the same project but this is the longer tem objective.

Where the appendices in the first edition continue to be relevant, they have been retained. Some new appendices have been added and others removed as being no longer relevant or appropriate. In particular, the preferred notation for process modeling of IFC projects has changed and a Readers Guide to the new notation has been included. It should be noted that the process model used to describe project development in this Guide uses the same notation and can therefore be used as an example of the approach.

Important additions to this edition of the Guide are appendices that outline the conventions used for development of the integrated IFC Object Model and that provide guidance on the model used for the specification of properties. Both of these are vitally important topics in IFC development.

As the work of the IAI progresses, changes are made to the structure of the organization to support better the development of projects. This means that information given in this Guide about the organization can only be accurate at a particular point in time. For this edition of the Guide, the organization structure is accurate at December 31st 1998.

We believe that this edition of the IFC Specification Development Guide will provide a better picture of the work required and the milestones associated with project development.

JDW

15-Mar-99

[1] The first version was titled 'Guide to the Specification of Industry Foundation Classes

This document gives guidance to members of the IAI on the methods to be used in the definition and specification of Industry Foundation Classes. It is issued in conjunction with Release 2.0 of the Industry Foundation Classes and contains guidance relevant to specification development for Releases 3.0 and above.

References within this document to the masculine gender shall be taken to imply equally the female gender and vice versa.

Use of the word:-

‘*programme’*	refers to a schedule of actions
‘*program’*	refers to a sequence of executable instructions to a computer
‘*project’*	refers to an agreed programme of work for incorporation in the IFC Object Model
‘*specification’*	refers to project information for incorporation into a model
‘*model’*	refers to a formal statement of classes, attributes, properties and behaviors that can be used to inform software implementation. An IFC model is a statement of requirements for structuring of information exchange and sharing.

Examples shown in the Guide are printed on a grey background.

Scope

The scope of this document includes the development of projects for the specification of Industry Foundation Classes for the AEC/FM industry including the following topics:

- scope and context definition;
- process analysis;
- identification of usage requirement;
- project specification development;
- specification review;
- test case development.

The following topics are not within the scope of this document:

- detailed information on conceptual languages for process, information, object and interface modeling[2;]
- guidance on software implementation;
- guidance on conformance testing;
- specification of object models for industries other than AEC/FM[3]

It is assumed that the reader is familiar with terminology in normal use within the AEC/FM industry.

[2] Readers guides to major information modeling methods used by the IAI are included. However, these are not detailed references and should not be treated as providing comprehensive descriptions.
[3] Although the scope statement excludes the development of object models for other than AEC/FM, the content may be relevant as a basis for specification development guidelines for other industry sectors.

Contents

Figures

Appendices

1. Introduction

The purpose of this document is to provide guidelines for the definition and development of Industry Foundation Classes. It provides guidance on the various stages of development to AEC/FM industry experts who may not be familiar with formal software development methods.

1.1. Related Documents

- Introduction to the IAI and the Industry Foundation Classes.
- IFC Release 2.0 Specifications

1.2. Background

The objective of sharing information between computer applications in the AEC/FM industry has been the target of research and development effort for a long time. The hardware and software technology that allows us to achieve this objective is now available. However, the greatest move forward has been in the development of tools and techniques that allow the inclusion of the users of the technology into the development process.

The open sharing of information without regard to the hardware or software applications in use is called interoperability. It emphasizes the value of information and how it is used rather than the systems which use it.

Interoperability requires that concepts that are common between different software applications be understood as common and declared accordingly. This understanding needs to be present within the computer systems running interoperable software and not just by their human operators. Since computer systems do not have the power of interpretation expected of a human user, development of the necessary understanding means that:

- names given to classes and properties correspond between applications;
- meanings assigned to these names are consistent between software applications;
- sharing of classes and properties between applications is unambiguous;
- relationships defined between the classes (including inheritance relationships) are compatible.

Concepts of interoperability must be developed in collaboration. End-users and software developers must work together in development. It is for these reasons that Industry Foundation Classes are being developed by the IAI.

2. Standards in IFC Specification and Use

The following commercial, national and international standards are employed in the specification and use of the Industry Foundation Classes:

Reference:	FIPS Pub 183
Title:	Integration Definition for Function Modeling
Publisher:	Computer Systems Laboratory, National Institute of Standards and Technology, Gaithersburg, Md. 20899, USA
Date:	December 21, 1993
Used for:	Development of Process Models using the IDEF0 graphical notation

Reference:	ISO 10303 Part 11
Title:	Implementation Methods; EXPRESS Language Reference Manual
Publisher:	International Standards Organization, Geneva, Switzerland
Date:	1994
Used for:	Formal specification of the IFC Object Model in the EXPRESS language

Reference:	ISO 10303 Part 21
Title:	Implementation Methods; Clear Text Encoding of the Data Structure.
Publisher:	International Standards Organization, Geneva, Switzerland
Date:	1994
Used for:	Syntax of the file structure used for data exchange according to the IFC Object Model

Reference:	ISO 10303 Part 22
Title:	Implementation Methods; Standard Data Access Interface.
Publisher:	International Standards Organization, Geneva, Switzerland
Date:	1994
Used for:	Syntax of the interface to databases that enable sharing of information according to the IFC Object Model.

Reference:	BS6100
Title:	Glossary of Building and Civil Engineering Terms
Publisher:	British Standards Institution. The compendium edition published by Blackwell Scientific Publications is used. Ref: ISBN 0-632-02851-3
Date:	1993
Used for:	Primary resource for the definition of terms.

Reference:	CORBA
Title:	The Common Object Request Broker Architecture and Specification: Revision 2.0
Publisher:	The Object Management Group Inc., Framingham, MA
Date:	July 1995
Used for:	Formal specification of software interfaces on the IFC Object Model in the Interface Definition Language (IDL).

Other standards are employed in the specification and use of particular parts of the IFC Object Model. Reference to these standards is made in the IFC Release documentation relevant to that part of the Model.

3. IAI Organization

Since its formation, the IAI has evolved an organizational structure that has three key elements. These are respectively, Chapters, Projects and International. The following defines the technical working parts of each of these elements.

3.1. Chapters

3.1.1. Domain Committees

Each Chapter has established a series of "domain committees" that are best suited to their representative members. A domain committee is interested in a specialized discipline, such as architecture or HVAC, or a specialized process, such as construction or facilities management. A domain committee is composed of members with experience in the area of the domain's specialization.

A domain committee is chaired by a domain expert who works with the Technical Co-ordinator of the Chapter to ensure proper representation of the Chapter requirements on the International Technical Management Committee.

A domain committee works within one or more projects to ensure that the specifications developed meet their national requirements.

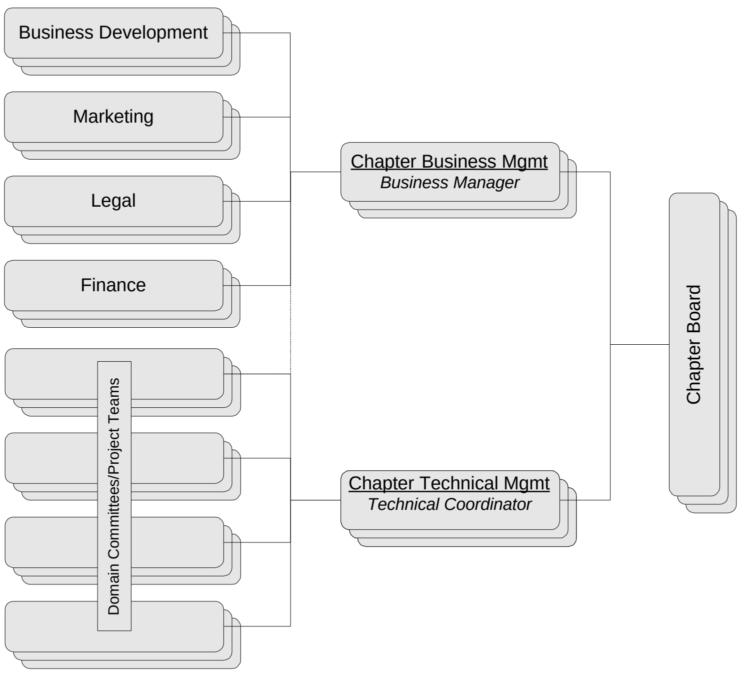


Figure 1 - Chapter Organization

3.2. Projects

IFC specifications are developed in Projects with each Project focussing on the satisfaction of one or more business requirements. Participants in a project contribute their expertize, allowing their knowledge of the information sharing needs of the business requirement to be formalized within an IFC specification or by reviewing and validating the work of other experts participating in projects. Expertize within a project can be classified under three headings:

- AEC/FM experts define the requirements;
- technical experts formalize and integrate the requirements into the specification;
- software vendors implement the requirements.

Participants in several Chapters normally undertake an IFC Project. Participants will also be members of a domain committee within a Chapter (see below)

Each Project must provide the financial and human resources necessary for its completion. This may be achieved by:

- contribution of human resource from member companies;
- the provision of development funds that can be used to recruit specialists;
- a combination of the above.

Each Project designates a Project Leader who acts as its representative on the International Technical Management Committee.

3.3. International

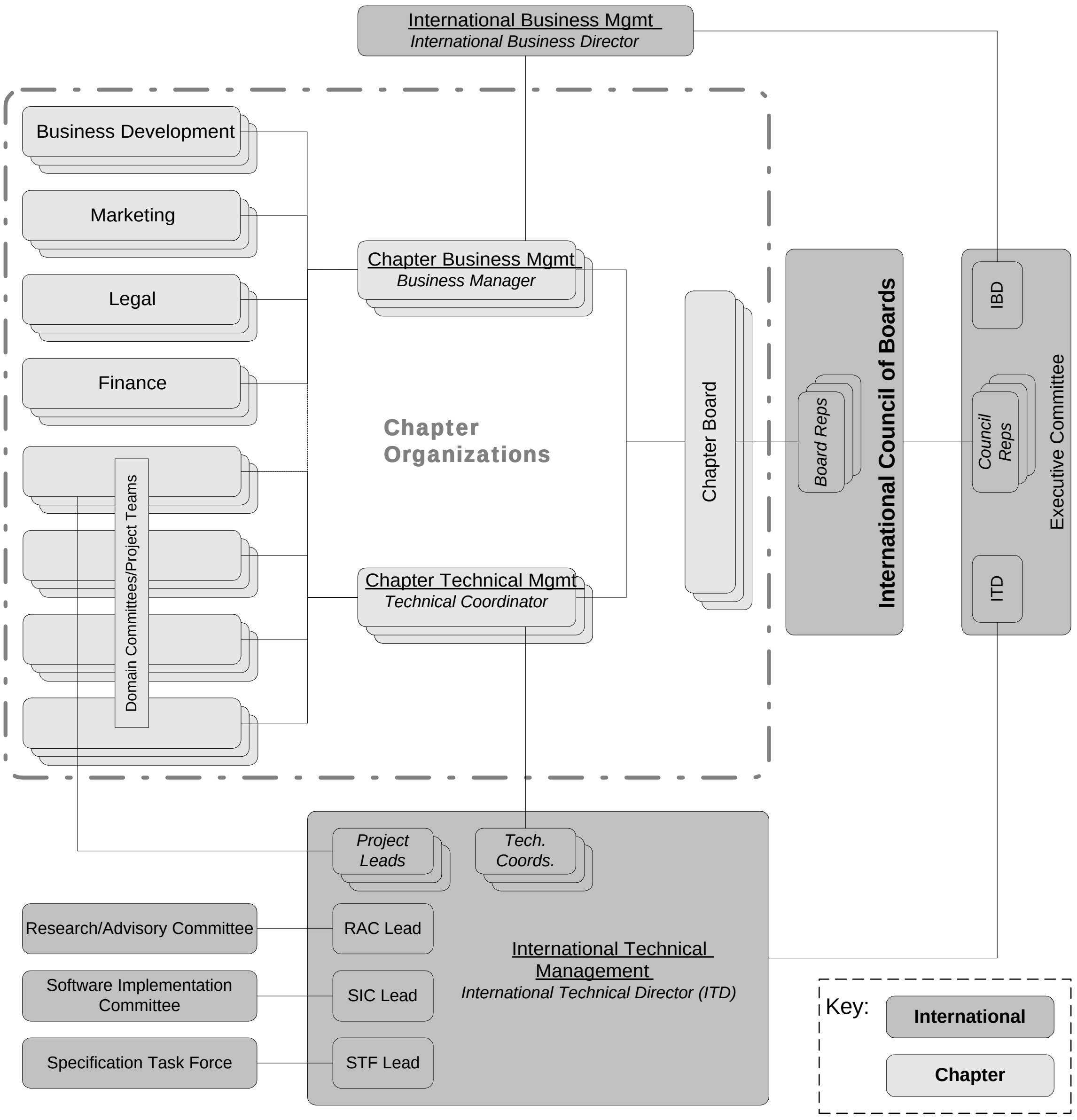


Figure 2 - International Organization

3.3.1. International Technical Management Committee

The International Technical Committee (ITM) is the principal technical body within the IAI. All other technical committees are represented on the ITM. Its responsibilities are to:

- provide the technical planning and management required for IFC development;
- coordinate international technical work;
- monitor/review/approve the IFC specification process and deliverables.

3.3.2. Specification Task Force

The Specification Task Force (STF) provides high level technical support to IFC Projects, Domain Committees and software implementers. The following tasks are the responsibility of the STF:

- develop the technical architecture of the IFC Object Model;
- develop or oversee the development of of domain model specifications;
- integrate IFC specifications into a single object model;
- review AEC/FM domain project requirements;
- develop the IFC Release document suite;
- support software implementers.

3.3.3. Software Implementation Committee

The Software Implementation Committee allows software implementers to work together in a spirit of open, pre-competitive collaboration to ensure that the IFC Object Model can be implemented in practice. Software implementers are responsible for:

- pilot implementation of the IFC Object Model;
- realization of commercial implementation according to individual company business plans;
- participation in AEC/FM domain projects defining requirements for IFC;
- provide input for the design/specification of the IFC Model.

Software implementers have a particular responsibility to the IAI in providing both the pilot and commercial implementations of IFCs. It is only through such implementations and the demonstration of their capability that IFCs will be deployed throughout industry and that support by industry of the IAI will grow.

3.3.4. Research Advisory Committee

The Research Advisory Committee (RAC) provides guidance to the IAI on technical matters including advances in technology that will affect the future development of IFCs. Members of the RAC are invited individually on the basis of eminence in their field of work.

Members of the RAC are invited to review and provide feedback on the IFC model design and all documents produced by the IAI. Such review tests the technical validity of the work when compared to the current ‘state of the art’ in the development of advanced information technology for use in the AEC/FM industry.

4. Overlapping Development

A primary goal of the IAI is to produce a new IFC Release annually with each Release extending the capability of the overall IFC Object Model. To achieve this goal, three distinct streams have to occur at the same time with each stream related to a different Release.

- At a given point in time, software implementers are developing products for IFC Release X.
- At the same time, the Specification Task Force are integrating the work of projects to deliver the IFC Release X+1 Object Model.
- Concurrently, the projects being undertaken by domain experts within the Chapters are related to the IFC Release X+2 Object Model.

By separating the work into streams, the 2-3 year period required for the total work of developing an IFC Release can be compressed into an annual release cycle. Necessarily, it means that projects will not see the results of their work appearing in a formal Release immediately and are will not see that work appearing in commercial software for several months after the issue of the relevant IFC Release.

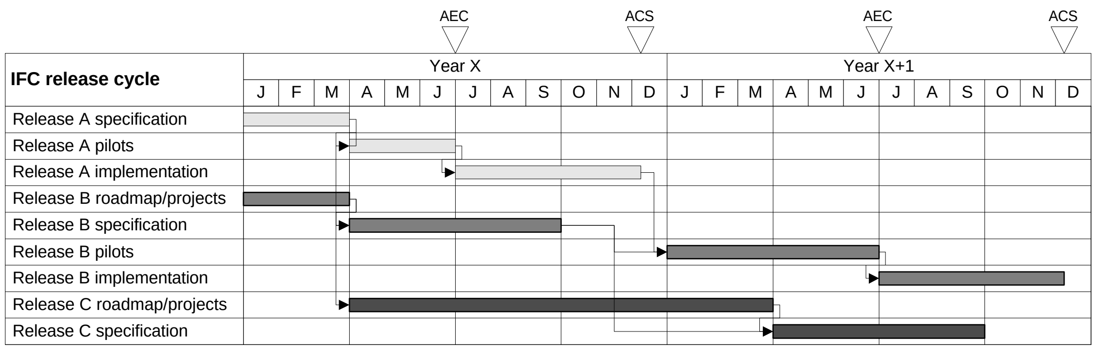


Figure 3 - Overlapping Release Development

5. Feedback During Development

There will be feedback between stages and substages during specification development. Development is not a strictly linear process. As work progresses, you will discover more about the process you are developing, more about the information requirements and more about the way the process is undertaken. This should be reflected back into work already done, refining it to provide a better solution. It is a spiral process that includes the need to test, validate and review after each step.

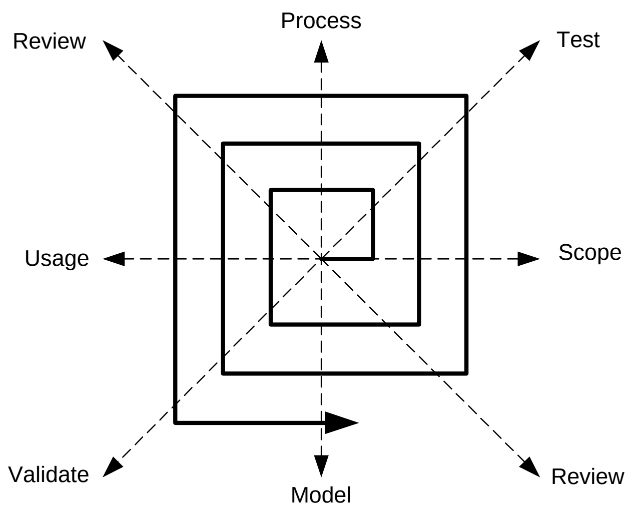
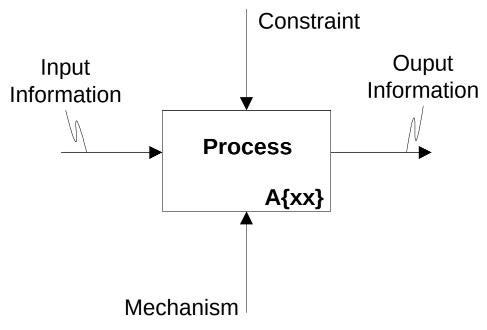


Figure 4 - Feedback during development

6. The IFC Specification Development Process

The IFC specification development process has the objective of understanding business requirements for information exchange and/or sharing and developing an object model and software implementations that can be used by practitioners within the industry to satisfy those requirements. It is a process that comprises a number of tasks each of which relies on information being supplied to it by other tasks.

The specification development process in this Guide is developed as an IDEF0 process model with the tasks being broken down progressively to enable information to be provided on what is required at each stage of the work. For more detailed information on the IDEF0 process modelling notation, refer to the IDEF0 Readers Guide.



Each individual process is shown in a box that contains the name of the process and a unique alphanumeric identifier. Since the process is active, it is always named as a verb phrase.

Input information enters at the left-hand side of the process box and output information leaves from the right-hand side of the process box.

Constraints that must be satisfied by the process enter at the top of the process box.

Figure 5 - Process model elements

Arrows entering the process box from below identify mechanisms, that is the actors that participate in the performance of the process. Generally, mechanisms shown in the process model for IFC specification development are the IAI teams that do, review or accept work. Therefore, in this Guide, mechanisms are referred to as Performers.

For simplicity, the full IDEF0 notation is not used in this Guide. In particular:

- the use of abbreviated coding to identify inputs, outputs, constraints and mechanisms is not used; this means that all arrows on al diagrams have a descriptive label.
- the use of tunneled arrows at high or low level to reduce the content of other diagrams is not used; this means that processes in all diagrams show all of the inputs, outputs, constraints and mechanisms that apply.

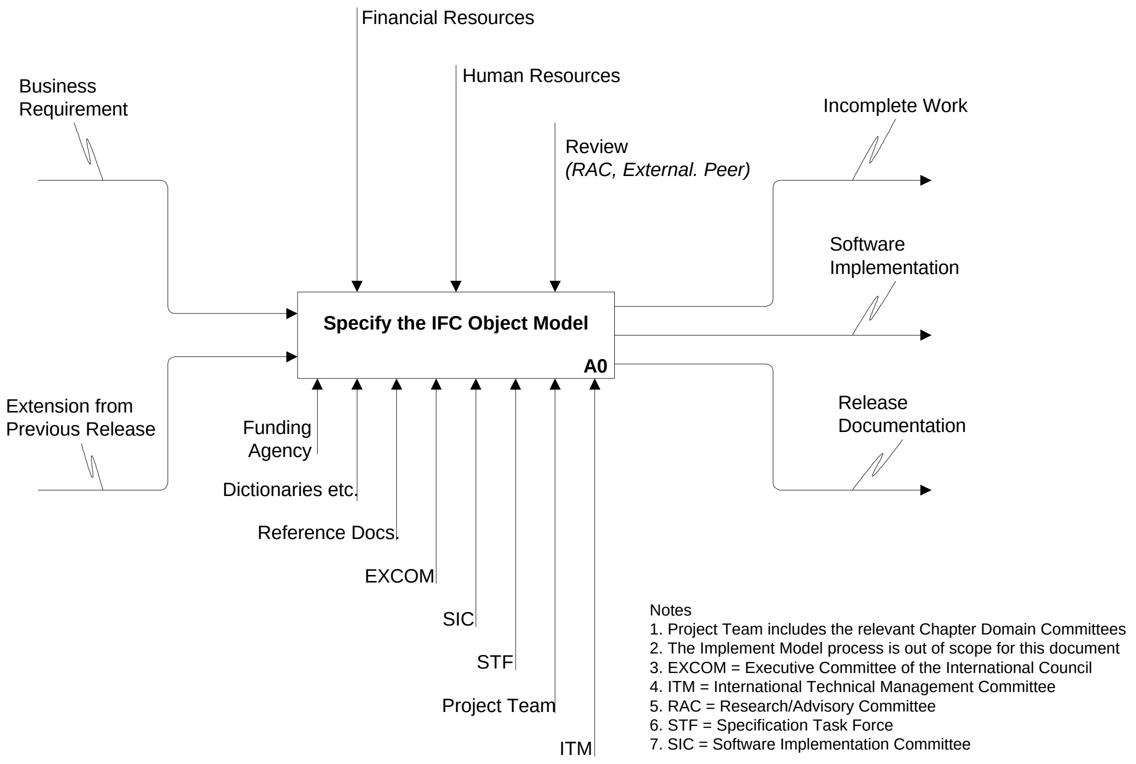
Where an identifier is shown below a process, this means that the process is further decomposed in an additional diagram. Each decomposed process diagram has the same label as its parent process. This means that decomposed processes have the identifier:

A + {Parent id} + {Child id}

Thus, a parent process A11 may have child processes A111, A112, A113 etc.

6.1. The Top Level Process

The top-level process model contains the key process which is named 'Specify the IFC Object Model' and which has the identifier A0. This is the overall process with which this Guide is concerned. It identifies the inputs that initiate the IFC specification development process and the outputs that finally result from the IFC specification development process. In addition, all of the constraints that affect the process and all of the performers that participate in the process are identified at this level.



Notes
1. Project Team includes the relevant Chapter Domain Committees
2. The Implement Model process is out of scope for this document
3. EXCOM = Executive Committee of the International Council
4. ITM = International Technical Management Committee
5. RAC = Research/Advisory Committee
6. STF = Specification Task Force
7. SIC = Software Implementation Committee

Figure 6 - A-0 Top level process model

Inputs	
Business Requirement	*Identifies the Business Requirement that causes an IFC specification development project to occur.*
Extension from Previous Release	*Where a project is an extension to a previously undertaken IFC specification development project, this constitutes a legitimate business requirement.*
Outputs	
Incomplete Work	*Incomplete work refers to an IFC specification development project that, for some reason, has not been completed.*
Release Documentation	*Identifies the complete set of documentation that accompanies an IFC Release.*
Software Implementation	*Refers to software that is written to satisfy the business requirement as expressed in the IFC Object Model*
Constraints	
Financial Resource	*Identifies the available financial resource that may constrain the extent of work that may be done by an IFC specification development project.*
Human Resource	*Identifies the available human resource that may constrain the extent of work that may be done by an IFC specification development project.*
Review	*Refers to the review processes that are undertaken at various stages of an IFC specification development project and that may constrain the work that can be completed for technical or other reasons. Particular reference is made to the Research/Advisory Committee (RAC) of the IAI that undertakes critical review of the IFC Object Model at various points in its development.*
Performers	
Dictionaries etc.	*Documents from which the meanings of terms (or semantics) may be obtained.*
EXCOM	*The Executive Committee of the International Council of the IAI that is responsible for executing the policies and resolutions of the Council on a day to day basis.*
ITM	*The International Technical Management Committee of the IAI which provides the forum for technical discussion on projects and that is responsible for decisions concerning the technical viability of a project.*
Project Team	*The team of people undertaking an IFC specification development project.*
Reference Documents	*Documents that provide sources of knowledge to a project.*
SIC	*The Software Implementation Committee of the IAI that represents the collective views of organizations writing IFC compliant software.*
STF	*The Specification Task Force of the IAI that provides overall technical guidance to projects and that is responsible for integrating the work of all projects into the single IFC Object Model.*

More detailed information about the role played by inputs, outputs, constraints and performers during the individual processes in specification development are given in the sections concerning individual processes.

6.2. Process A0 - Specify IFC Object Model

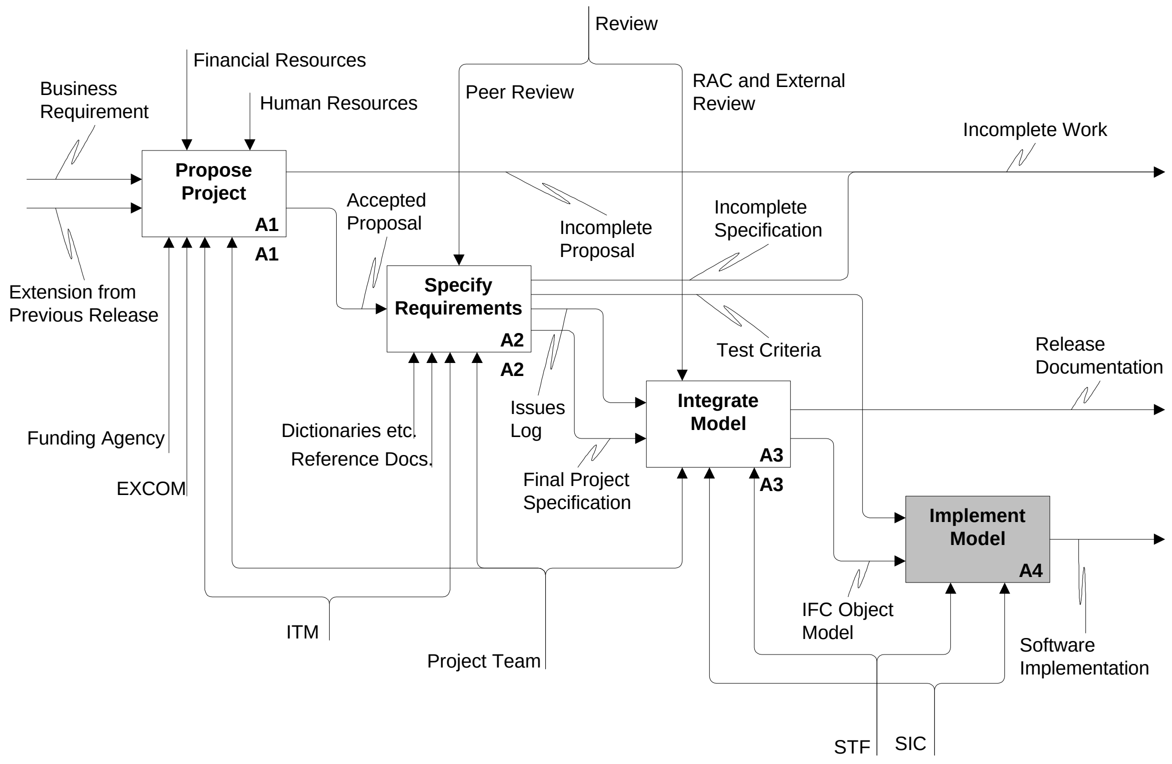


Figure 7: A0 Specify the IFC Object Model

This process identifies the principal stages in the development of an IFC specification namely:

There are five sub-processes involved in the preparation of a project proposal leading to its acceptance for inclusion within an IFC Release programme or its identification as incomplete:

A11. Propose project
A12. Specify requirements
A13. Integrate model
A14. Implement model

Implementation of the model is not within the scope of this document. However, the sub-processes that constitute the overall process include:

- Pilot implementation of the released IFC Object Model.
- Conformance testing and certification.
- Commercial implementation and shipping of certified IFCs by software vendors.
- Feedback from customers for maintaining and improving IFC releases.

6.2.1. Process A1 - Propose Project

Before starting to work on the development of an IFC specification, a proposal needs to be provided to the ITM of the intended development. The need for the proposal is to ensure that:

- the intended development fulfils an identified industry need;
- Whilst an industry need may be identified within one Chapter, the objective is to seek out industry needs across all Chapters so that IFCs can support global interoperability.
- the intended development provides a coherent extension to the current IFC capability;
- The ITM encourages the development of specifications that promote interoperability across domains. Therefore, the ITM actively looks for industry processes that can operate in conjunction with each other rather than processes that exist in isolation.
- the resource ensuring complete development of the specification is available;
- The ITM will assist the development of a proposal by seeking additional resources to support development from Chapters other than that initiating the proposal.

A template document for completing the proposal is provided which enables assessment of the proposal in conjunction with others. It is designed to assist continued IFC development by:

- enabling the ITM to consider all proposals on a consistent basis;
- enabling the project team to build progressively on information provided within the proposal.

The template is available from the IAI FTP site.

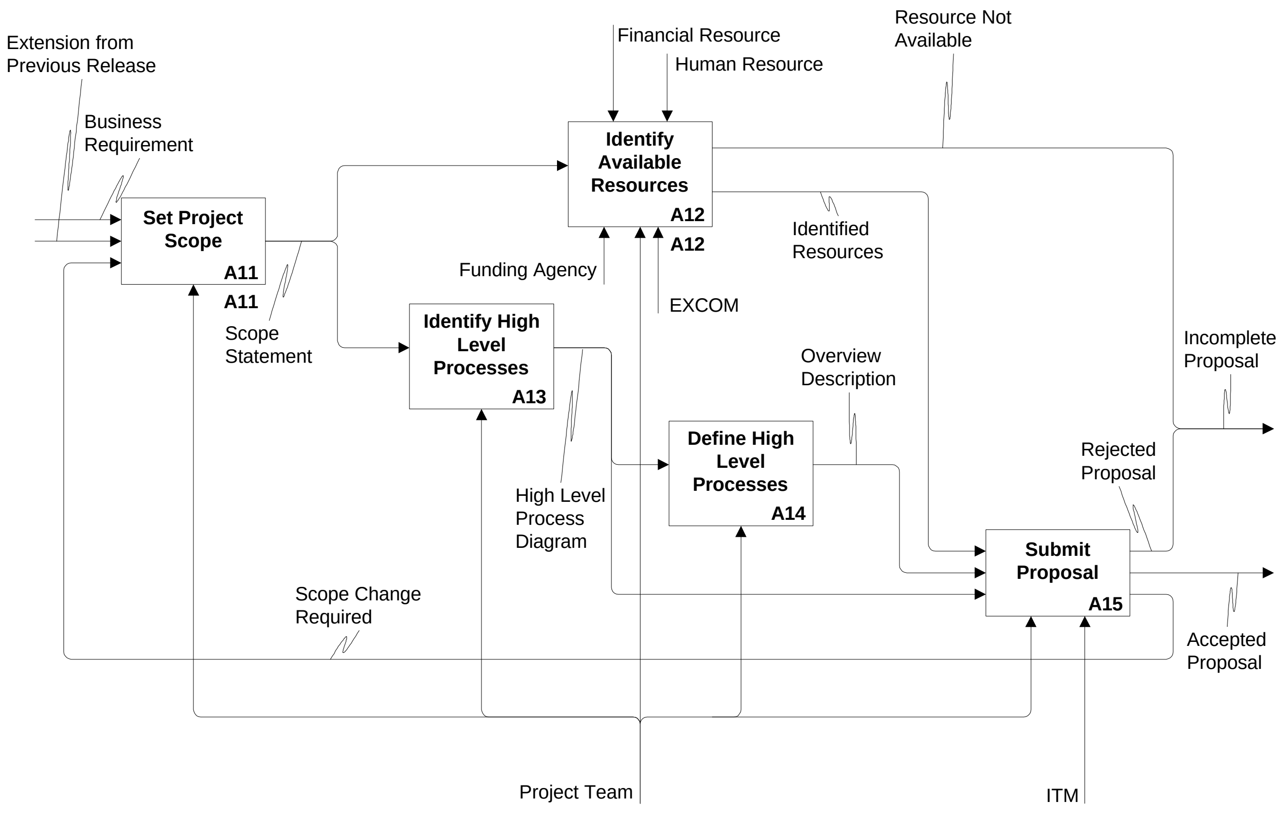


Figure 8: A1 Propose Project

The objective of the proposal document is to ensure that:

- there has been sufficient consideration of the effort required in completing the work within the required timescale (including both human and financial resource considerations);
- the amount of work required to complete a proposal is kept to a minimum.

There are five sub-processes involved in the preparation of a project proposal leading to its acceptance for inclusion within an IFC Release programme or its identification as incomplete:

A11. Set project scope
A12. Identify available resources
A13. Identify high level processes
A14. Define high level processes
A15. Submit proposal

6.2.1.1 Process A11 - Set Project Scope

The scope statement provides an early indication of the work requirement for a project. It sets the boundaries for the work that is to be done and provides a continuing reference to ensure that the work boundaries do not grow beyond a point at which the planned or available resource ceases to be sufficient.

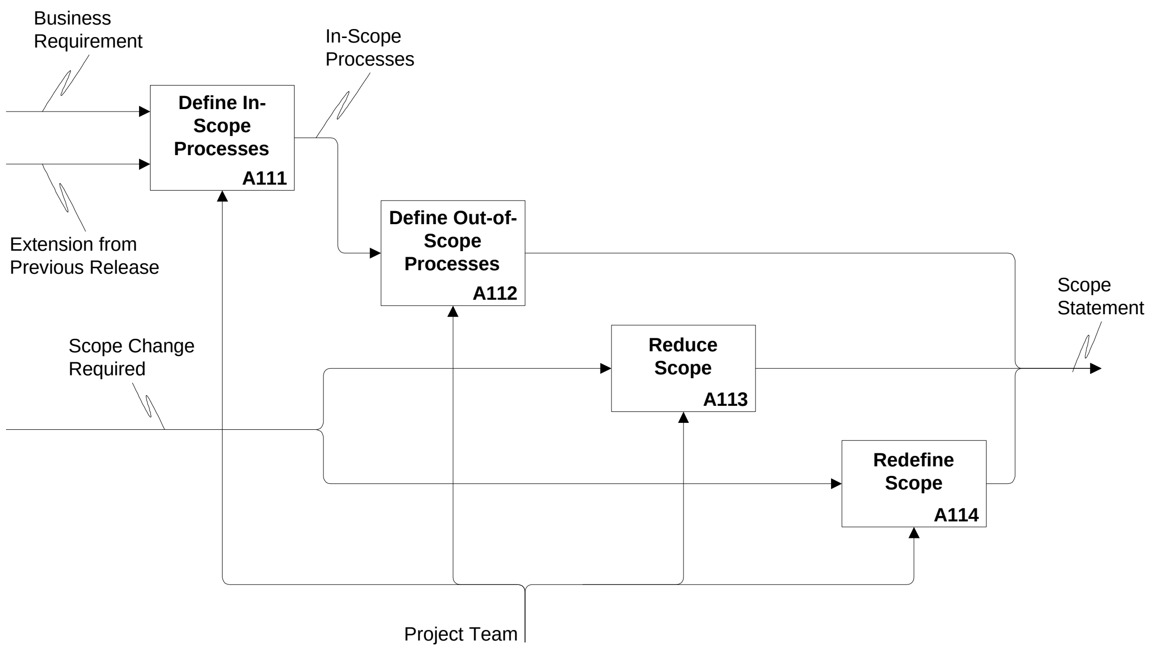


Figure 9: A11 Set Project Scope

6.2.1.1. Process A111 - Define In-Scope Processes

Inputs	
Business Requirement	*Identifies the Business Requirement that causes an IFC specification development project to occur.*
Extension from Previous Release	*Where a project is an extension to a previously undertaken IFC specification development project, this constitutes a legitimate business requirement.*
Outputs	
In-Scope Processes	*Defines those processes that are in scope of the project.*
Constraints	
-	
Performers	
Project Team	*Define the project scope.* *Caution is advised when defining scope to ensure that the project can be completed within the time scale set. Obtaining broad international agreement to the processes and classes within a project will consume more time and resources than for an object model defined for local use.* *Do not be too ambitious. A small scope that serves a useful purpose and that can be completed may be more valuable than a large scope with its inherent complexity.*

6.2.1.1. Process A112 - Define Out-Of-Scope Processes

Inputs	
In-Scope Processes	*Input from A111 above*
Outputs	
Scope Statement	*Defines the complete scope statement for the project including processes that are in-scope and those that are out-of-scope.*
Constraints	
-	
Performers	
Project Team	*Identify processes that are out of scope for the current project.*

6.2.1.1. Process A113 - Reduce Scope

Inputs	
Scope Change Required	*A reduction in scope may be requested by the ITM. The aim is to define that portion of a proposed development that can be completed successfully within a development cycle.*
Outputs	
Scope Statement	*A modified scope statement in which the scope has been reduced to meet criteria defined in the scope change request.*
Constraints	
-	
Performers	
Project Team	*Perform scope reduction*

6.2.1.1. Process A114 - Redefine Scope

Inputs	
Scope Change Required	*A redefinition in scope may be carried out by the project as a result of further knowledge gained in specification development.*
Outputs	
Scope Statement	*A modified scope statement in which the scope has been redefined to meet criteria defined in the scope change request. Where redefinition results in substantial reduction or expansion in scope, the collaboration of the ITM should be sought. This is so that the impact of scope change on any other projects can be properly assessed.*
Constraints	
-	
Performers	
Project Team	*Perform scope redefinition.*

Example:

The following defines the scope for the Plan Maintenance process within the domain of Engineering Maintenance. It states the activities that are involved in maintenance planning and those activities that are bot dealt with by this process.

Engineering Maintenance consists of several processes. Each process has a separate scope.

Process Scope:

- Identify Assets To Be Maintained
- Identify Maintenance Action
- Assign Action To Asset
- Prepare Scheduled Work Order
- Schedule Maintenance

Out-of-Scope:

- Prepare Work Order for a specific maintenance task
- Record maintenance history
- Consider inventory of tools and equipment for maintenance

6.2.1.2 Process A12 - Identify Available Resources

A successful project is one that can deploy the necessary resources to achieve its scope requirements within the appropriate timescale. Insufficient thought to the question of resource availability can result in failure to deliver. This is disappointing not only to the project team concerned but can also affect other work that relies on completion of the project.

The proposal documentation includes a section that includes expected resources to be identified and the shortfall determined so that additional funding can be sought.

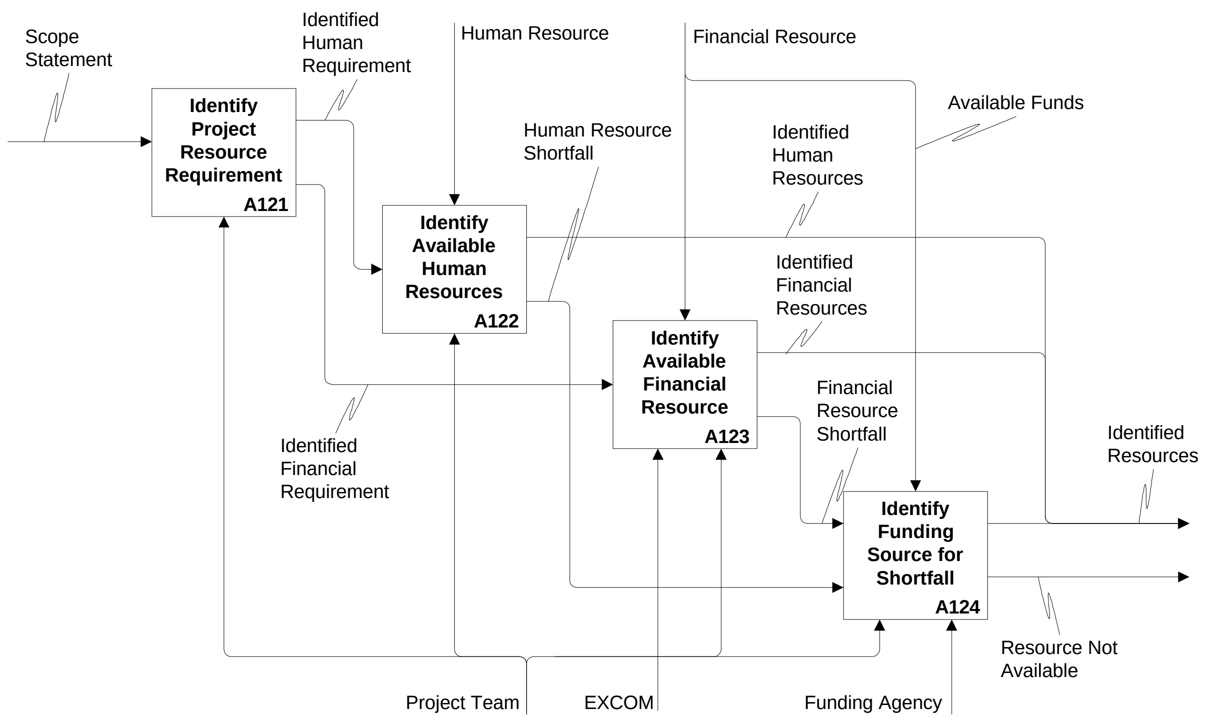

Scope Statement
Identified Human Requirement
Human Resource
Financial Resource
Available Funds
Identify Project Resource Requirement A121
Human Resource Shortfall
Identified Human Resources
Identify Available Human Resources A122
Identified Financial Resources
Identify Available Financial Resource A123
Financial Resource Shortfall
Identified Resources
Identified Financial Requirement
Identify Funding Source for Shortfall A124
Resource Not Available
Project Team
EXCOM
Funding Agency

Figure 10: A12 Identify Available Resources

6.2.1.2. Process A121 - Identify Project Resource Requirement

Inputs	
Scope Statement	*The scope statement allows the extent of the project to be assessed*
Outputs	
Identified Human Requirement	*Identifies the total time requirement for people who will be participating in the project.*
Identified Financial Requirement	*Identifies the total financial requirement of the project without regard to the time that may be contributed by project participants. Identified finacial requirements should include:* • *the total cost requirement of domain expertize;* • *specialist software that the project might use;* • *expenses that might be given to project participants for meeting attendance;* • *the cost of hiring IT specialists to support the project.*
Constraints	
-	
Performers	
Project Team	*Identify the total financial and human resources required to complete the project according to the defined scope.*

6.2.1.2. Process A122 - Identify Available Human Resources

Inputs	
Identified Human Requirement	*See A121 above*
Outputs	
Identified Human Resources	*Identifies the people who will be working on the project and the amount of time that each person can contribute. The Chapter affiliation of the participants should be quoted so that the international coverage of the project can be seen.* *It should be recognized that time for both project domain experts and IT specialists should be considered. IT specialists involved in creation of the data model may need to be funded because of the specialist nature of their work.* *Domain expertize will normally be provided by people working in industry who have extensive responsibilities outside of the IFC specification development project. Therefore, the amount of time that each person in the project team can contribute on a per week or per month basis may be limited. A challenge for any project is how to use the available time, in what is largely a volunteer effort, to maximum effect.* *Proposers should seek the support of software implementers to identify that, as well as an industry interest, there is also the probability that software implementations will be developed. Software implementers can advise on the scope of the proposal so that both the specification and its software implementation can be completed in time for the target Release.*
Human Resource Shortfall	*The difference between the human resource requirement and the identified human resources constitutes the human resource shortfall. If the project is to proceed and fulfil its objectives, this shortfall needs to be addressed and additional resources provided.*
Constraints	
-	
Performers	
Project Team	*Identify the people who will participate in the project.*

6.2.1.2. Process A123 - Identify Available Financial Resources

Inputs	
Identified Financial Requirement	*See A121 above*
Outputs	
Identified Financial Resources	*Identifies the finances that are available to the project without having to seek external funds. It includes the costed value of human resources identified in A122 above together with any additional financing that may be offered by project participants.*
Financial Resource Shortfall	*The difference between the financial resource requirement and the identified financial resources constitutes the financial resource shortfall. If the project is to proceed and fulfil its objectives, this shortfall needs to be addressed and additional resources provided.*
Constraints	
Financial Resource	*The financial resource that is available will constrain the scope of the project.*
Performers	
Project Team	*Locate the financial resources that are available within the project. This requires identification of the time an/or money that will be contributed by project participants.*
EXCOM	*Part of a project is the need to ensure that it is integrated with the work of other projects to deliver the IFC Object Model. Part of the project finance therefore should enable a resource to be dedicated to this task. Assistance will be provided from the Specification Task Force at the stage of integration but the extent to which this will be available depends on the funding that can be allocated to this task by the Executive Committee of the International Council (EXCOM). Therefore, EXCOM is a performer in the task of identifying financial resources and should be included in decision making.*

6.2.1.2. Process A124 - Identify Funding Source for Shortfall

Shortfalls in funding may be made good by obtaining additional finance from external funding agencies. These may be private companies, a consortium of companies or public research funding programs.

Normally, requests for funding need to be made in a particular format. Programs may impose limits on the amount of money that may be made available for asingle project or limit the percentage of the total that they are prepared to fund. Proposals for funding may have to be made at a specific time and there may be a delay before decisions are taken about which projects are to be funded (in cases where competitive criteria apply) and when the funding may become available.

All of the above points need to be taken into account when identifying the funding source to overcome a shortfall in resources available to a project.

Inputs	
Human Resource Shortfall	*See A122 above*
Financial Resource Shortfall	*See A123 above*
Outputs	
Identified Resources	*Identifies the total resources that are available for a project including any additional resources that have been granted to the project.*
Resource Not Available	*If a request for additional funding for a project is not successful, the project team may determine that continuation with the project is not feasible since the required resource is not available. Alternatively, scope reduction or redefinition may be required to match expectations with available resource.*
Constraints	
Available Funds	*Funds available from funding agencies are often constrained to a maximum amount or may be limited to a maximum percentage of the total funding requirement for a project. These constraints need to be taken into account when making a request for additional funds to overcome a shortfall.*
Performers	
Project Team	*Identify the extent to which the available funds might fall short of the total amount that is required to complete the project.*
Funding Agency	*Identifies the funding agency that is providing the additional funds to overcome the shortfall in available resources.* *Funding agencies may impose requirements on the project as a result of providing funding. For instance, for a project that receives public sector funding, there may be a requirement that the project results all become publicly available. Conditions that may be imposed should be considered against the publishing policies of the IAI. Guidance should be sought from EXCOM on this matter.*

Example:

Project Team

Project Leader: Bruce Foster -- Aus

Chapter	*Name*	*Email*	*Hrs / Week*
Aus	Bruce Foster	bruce.foster@alice.springs.gov.au	12.5
NA	John Smith	john.smith@ourcompany.com	3.5
NA	Mary Tudor	mary@hamptoncourt.com	7
France	Roland Orlando	rolando@charlemagne.court.fr	3.5
France	Francois de la Valliere	f_valliere@champignon.fr	3.5
Germany	Bernd Uberbahnhof	bu@wagner.ring.de	3.5
Japan	Yoko Narita	yoko.narita@mizumi.nimail.jp	4
UK	Henry Fitzhenry	h.fitz@thecity.bysea.uk	3.5
	other		?
	Total for team	***(5 days per week)***	***41***
	Total person-days	***(person-days for 50 weeks)***	***255***

Scope of Work

# of AEC processes to be supported	3	Est. total AEC expert time (days)	5
Expected IFC Model Impact (1 *(min)* to 5)	4	Est. total Info Modeling expert time (days)	61.5
Degree of technical difficulty (1 *(min)* to 5)	3	Est. total Software/PM expert time (days)	32

Resources Required / Committed

Member Company Resources	**Required Days**	**Market Value (apply $80/hr-head)**		**Days Committed**	**Resource shortfall**
Requirements definition					
Process Model	20	$12,000		20	0
Usage Scenaria	45	$27,000		45	0
Model design					
Object Model development *(w/ tech.Support)*	30	$18,000		30	0
Integration *(w/ tech.Support)*	30	$12,000		20	10
Design and Implementation validation					
Test Case development	50	$24,000		30	20
Review/feedback on implementations	30	$12,000		10	20
Project Management					
Project management and administration	20	$12,000		20	0
Travel and Meetings	100	$48,000		80	20
Total Member Company Resources	**325**	**$165,000**		**255**	**70**

Model/Specification development support	**Required Days**	**Market Value**		**Days Committed**	**Resource shortfall**
Technical support	40	$24,000			40
Project management	20	$12,000			20
Publication and Administration	30	$18,000			30
Equipment and software		$12000			
Travel and subsistence		$12000			
Total Project Support	90	$38400			
Total for Project	365	$212400			160

Shortfall = 160 days @ 7.5 hours per day @ US$80/hr = $96,000

6.2.1.3 Process A13 - Identify High Level Processes

Inputs	
Scope Statement	*See A11 sub-processes above*
Outputs	
High Level Process Diagrams	*High level process diagrams are graphical models that are created (by IAI preference) in the IDEF0 notation. They identify the principal processes in a project and do not need to be fully broken down into sub-processes.* *For instance, the IFC specification development process is described in this manual using IDEF0 diagrams. The high level process diagram is that shown for process A0 – Specify Object Model identifying the high level processes as:* *A1. Propose Project* *A2. Specify Requirements* *A3. Integrate Model* *A4. Implement Model*
Constraints	
-	
Performers	
Project Team	*Create high level process diagram that identifies the principal processes that fulfil the business need of concern to the project.*

Example:

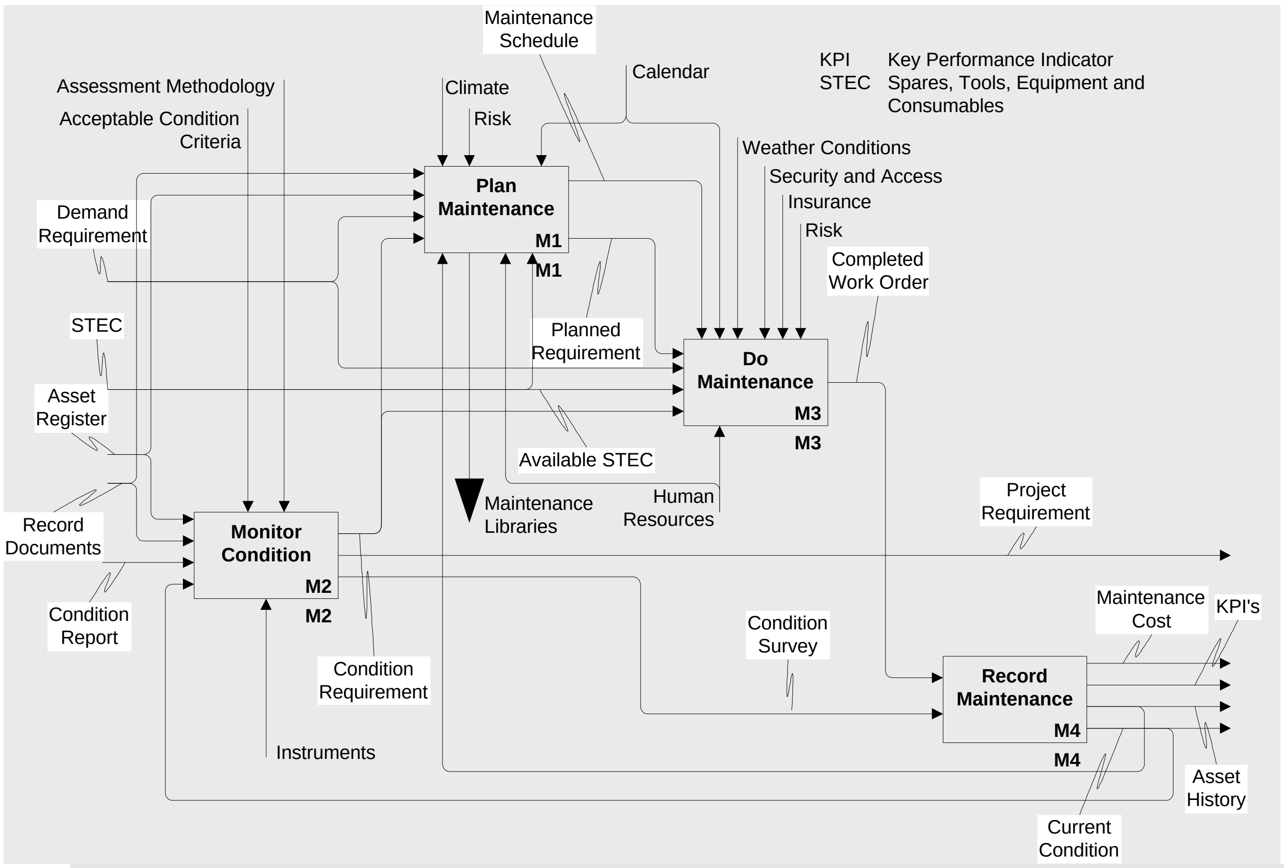


Figure 11: Example High Level Process Model for Maintenance

6.2.1.4 Process A14 - Define High Level Processes

Inputs	
High Level Process Diagram	*See A13 above*
Outputs	
Overview Description	*An overview is a textual description that expands the meaning of a process shown on the process diagram. Its purpose is to explain the intent of the process and why it is included in the project. Note that an overview of the whole process should be included and it is useful if this identifies features that the resulting model will offer and the benefits that can accrue from them. The features and benefits help to demonstrate the business requirement to others not involved with the project.* *If additional resource has been requested through a funding agency, it is probable that the features and benefits list will form part of the submission and can be reused at this point.*
Constraints	
-	
Performers	
Project Team	*Document the purpose of the processes identified.*

Example:

The example shown is the high-level process diagram for Plan Maintenance process. Note that in this example, the letter M prefixes processes; the character to be used is selectable (but avoid using O, and I where possible).

[FM-1] Engineering Maintenance

Engineering Maintenance comprises many processes. For present purposes, this document outlines the following processes:-

- Plan Maintenance including identification of the assets to be maintained, maintenance actions required and the scheduling of those actions.
- Monitor condition; in this case strictly the assessment of condition from visual inspection, the recording of condition and the determination of maintenance requirement from the condition assessment.
- Carrying out of maintenance operations
- Recording maintenance work to establish the life-cycle history of assets

An asset may be considered as something which contributes to the value of the organisation which owns it and consequently may, or may not, be subject to maintenance requirements. For the purpose of this process, it is assumed that only assets to be maintained are in scope.

Much of the information required by maintenance applications is available from the design and installation stages of a project. The ability to capture this information from design, estimating and management applications directly into an engineering maintenance application has many features and benefits including:

Feature	*Benefit*
Operating and Maintenance information can be captured during project design and construction stages.	Reduces cost of asset list creation.
	Reduces the requirement for inspection visits and information gathering by the owner/operator after project handover or by a maintenance contractor prior to taking on or taking over a maintenance contract.

Feature	Benefit
Greater certainty that operating and maintenance information has been fully captured as it will not be missed or hidden during inspection visits.	Reduces risk in delivering the maintenance service. Risk value may be as high as 10% on a new project and, for comprehensive maintenance on an existing project it may be from 15% to 40% depending on age, condition and other factors. It is estimated that with higher quality and more complete information, the risk value can be at least halved without affecting profitability for the maintenance contractor.
Details of items requiring maintenance are complete at project handover including item specification and supplier.	Reduces cost of order/invoice development for the client on shared risk contracts.
	Reduces recurring annual cost of 5% - 10% p.a. that normally applies due to not having all required data.
Complete and certain information with reduced cost of risk enables a maintenance contractor to provide an improved service at lower cost with equal or better profitability.	Increases the competitiveness of the maintenance contractor.
Access to operating and maintenance that can be directly incorporated into or referred to by maintenance management systems.	Accuracy of information will lead to better maintenance and improved maintenance scheduling.
Access to industry standard information that can be directly used for creating work orders.	Use of best practice information agreed on an industry wide basis will lead to improvements in the quality of maintenance work done.
	Best practice information will include for assessments on spares, tools, consumables and labour use that can lead to improved maintenance scheduling.
More complete and better information available during operation and maintenance.	More extensive queries and reports can be made that can lead to better analysis of the lifecycle and reliability of items being maintained.

Activity M1 - Plan Maintenance

Overview:

This process is concerned with acquiring sufficient information to enable planning and scheduling of the maintenance to be undertaken. Maintenance planning can encompass:

- periodic maintenance requirements that can be discovered from data available during the design and construction phases;
- maintenance requirements determined from condition monitoring and assessment where the need for maintenance is not immediately urgent and can be included within a planned requirement;
- maintenance requirements determined from demand requirements where the need for maintenance is not immediately urgent and can be included within a planned requirement;

Inclusion of condition and demand requirements would normally be on the basis that work can be scheduled to occur at the same time as a periodic maintenance activity on the same or an adjacent asset. Where condition or demand maintenance is considered to be urgent, it can still be dealt with directly by issuing a work order (see Do Maintenance).

Using the identities of equipment requiring maintenance from the asset register, the scope of this process is to identify the required planned maintenance actions at an early stage and to prepare scheduled work orders from which individual work orders can be derived as work is required. The generation of actual work orders and the carrying out of maintenance actions is outside the scope of this process.

The need to provide information from a maintenance system to a financial system is encompassed within this process. This is through the ability to specify a cost for the scheduled work order (which will be the estimated cost of carrying out work against a derived work order and not the actual cost resulting from the work undertaken against that work order).

Activity M2 - Monitor Condition

Overview:

This process is concerned with determining if the current condition of an asset is such that it requires maintenance in order to bring it to a required level of operating efficiency.

There are three possible scenarios that may result from monitoring the condition of an asset:

1. Maintenance is required in which case the work may be either –
   - sufficiently urgent as to warrant the issue of a demand work order
   - able to be carried out at some planned future point at which time work will be undertaken on the asset anyway or an adjacent asset.
2. The condition of the asset is such that it requires replacement in which case the work requirement may be subject to a project requirement.
3. Condition is satisfactory in which case no maintenance is currently required.

Monitoring the condition of an asset may be undertaken in two ways:

1. Instrumentation is in place that enables operating parameters to be continuously monitored; the parameters being selected such that they indicate the current condition of the asset (e.g. vibration). Where condition is continuously monitored, the point at which maintenance will be required will be determined by the values of the parameters being monitored exceeding a given value.
2. Condition of the asset is periodically monitored by inspection that may be either visual, carried out with the assistance of instruments or a combination. Where condition is periodically monitored, the point at which an inspection takes place may be determined either by –
   - a planned program of inspection, the inspection itself being considered to be part of a planned maintenance regime and subject to the issue of a work order;
   - a fault report being generated by a user of the asset, the fault needing to be inspected to determine its cause and whether maintenance is required.

At this stage, the process is concerned only with determining the condition of an asset by inspection and not by determining the condition of the asset through continuous monitoring.

Activity M3 - Do Maintenance

Overview:

The requirement is the actual creation and execution of a work order in response to a maintenance requirement. The requirement for maintenance may originate from various sources namely:

- Planned (within the planned preventive maintenance system).
  In this case, the resulting work order is an instance of a Master Work Order that is defined as a result of a previous process.
- Condition.
  This is similar in nature to the planned maintenance action in that the work order may be an instance of a Master Work Order. However, the execution of the work order is the result of the asset having reached a certain condition through usage or it having been operated for a set amount of runtime (as opposed to calendar time). Therefore, whilst the maintenance action may be planned, it may not be possible to associate it with a planned maintenance schedule. Issuing of the work order is as the result of inspection of condition or knowledge of parameters determining condition through instrumentation (e.g. vibration analysis indicating the condition of bearings).
- Demand.
  A demand maintenance action is carried out in response to a request for action from the users of the facility or from the occurrence of an unplanned event(s) affecting the satisfactory normal operation of the facility. Frequently, demand maintenance is executed through a Helpdesk facility which takes the request, logs it, ensures that it generates a required maintenance action and issues the necessary work order for the action.

It is necessary to be concerned with each of these types of work order since each plays a part in the establishment and operation of an engineering maintenance regime. Demand maintenance may be responsible for over 50% of the calls made on the maintenance capability.

The result of carrying out work orders is that the life of the asset is extended since the objective of maintenance is to return it to an optimal operating condition. Additional benefits occur from the information which is acquired whilst executing the work order. These include information for:

- maintenance history so that long term analysis and decision making is supported (see separate process);
- better control of spares;
- assessment of staff performance through time sheet provision and logging of time against work orders (which may be particularly relevant when using external maintenance contractors).

Activity M4 – Record Maintenance

Overview

All information the asset, its condition and work orders undertaken on it are recorded. This record or maintenance history enables the life-cycle performance of the asset to be tracked.

6.2.1.5 Process A15 - Submit Proposal

Inputs	
Identified Resources	*See A12 sub-processes above*
High Level Process Diagram	*See A13 above*
Overview Description	*See A14 above*
Outputs	
Rejected Proposal	*Submitted proposals that do not fully meet business needs, document completion, technical or financial criteria may be rejected from the current release.* *Normally, proposals are not rejected totally. Recommendations may be made that more preliminary development work is required on the proposal or that more resource is required and that it should be resubmitted for the next IFC Release. More often, it is returned to the project team with a scope change requirement.*
Scope Change Required	*Submitted proposals that do not fully meet business needs, document completion, technical or financial criteria may be returned to the project team with a scope change requirement.* *The most common reason for a scope change requirement is that the technical objectives of the proposal may, in the opinion of the ITM, be too ambitious for the timescale of the target IFC Release. In this case, a scope reduction is requested.*
Accepted Proposal	*Proposals that meet all submission criteria are accepted into the target IFC Release program. Therefore, work on development of the detailed project specification can proceed.* *When all proposals have been completed, reviewed and amended to take account of review comments, the selection of those processes which are to be taken forward to completion for a target Release can be finalized. This final acceptance of an IFC specification development project places the proposal onto the Release program. This program is then advertised by the IAI as the content that will be provided to industry at a particular time.* *Final selection of a process places obligations onto the project team:* • *to other members of the same Chapter who need to support the work;* • *to members of other Chapters who have agreed to support and review the work;* • *to organizations in the wider industry who are now expecting this capability and/or who may have become members of the IAI as a result of this expectation.*
Constraints	
-	

Performers	
Project Team	*Ensure that the proposal is technically complete for this stage of the work and submit to ITM.*
ITM	*The ITM reviews the proposal for its technical completeness, the assessment of resources to complete the work, the support that it has in other Chapters and its relationship with existing models and other proposed processes. The following criteria are applied:* *1. A proposal that is not technically complete will be returned to the authors with suggestions for further work. If it is not technically complete or if it is felt that it is not reasonable for the work to be completed within the release cycle under consideration, the ITM will suggest that the proposal should be developed further for the next IFC release cycle.* *2. The experience of the ITM and its related technical committees will be applied in assessing the resources shown for a proposal. If the resources shown are considered to be insufficient, the ITM will suggest that more are found or that the development takes longer.* *3. An assessment will be made of support from Chapters other than that preparing the proposal. Proposals having international support will be given preference. Where a proposal does not have international support, the ITM will assist in obtaining it. If international support cannot be raised, it will be considered that the proposal is of local interest only at that time. It will be suggested that the proposal should wait until a future release when further support may be forthcoming.* *4. A proposal will be assessed for its contribution to the development of the goal of interoperability. This will involve determining how it relates to current and other proposed specification developments. Proposals that can interoperate with existing or other proposed processes will be accepted more readily than those that appear to stand-alone.*

6.2.2. Process A2 - Specify Requirements

All of the tasks to be undertaken during the Specification Requirements process build on work already completed during the Propose Project process. Preliminary detail for process model, process overview, scope and resource requirement will be available.

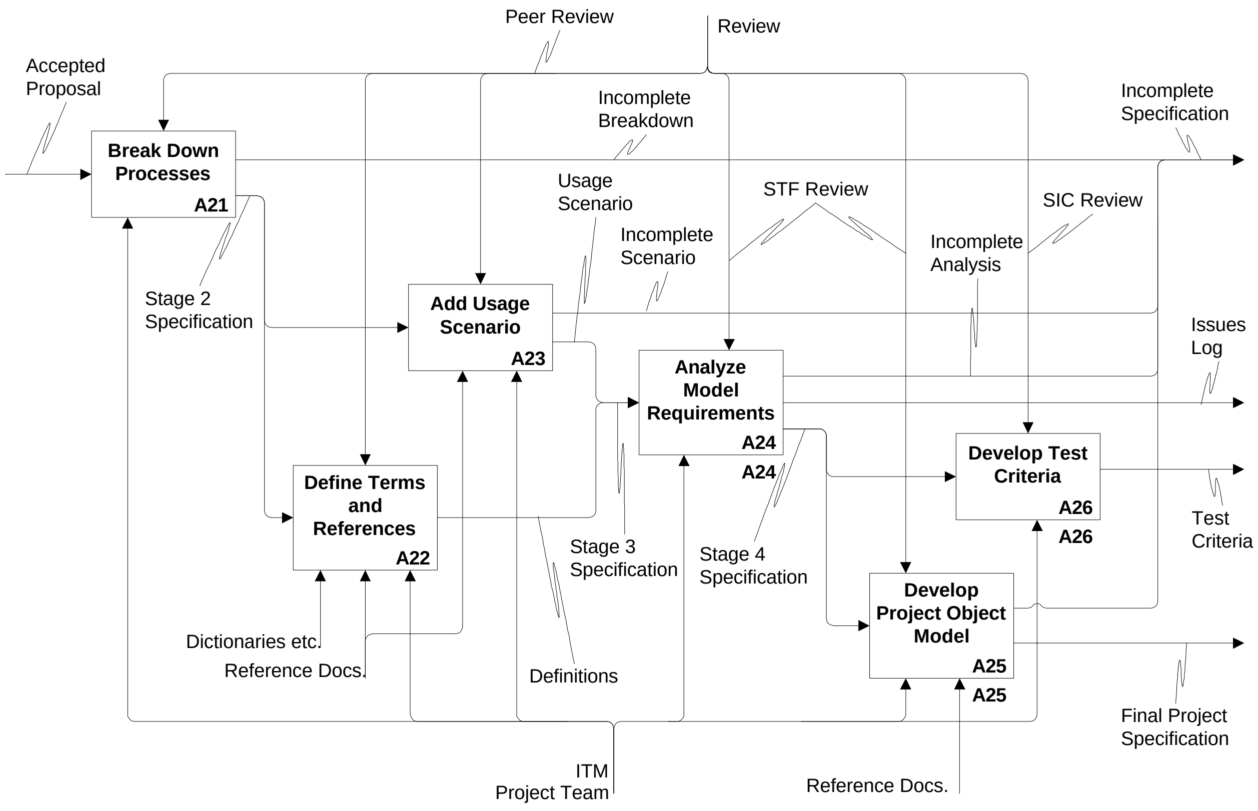


Figure 12: A2 Specify Requirements

The objective of the model specification is to ensure that:

- the process model that defines the business requirement is sufficiently developed to enable identification of the data supporting the processes;
- terms that are used within the process are defined so that their semantic meaning is not ambiguous;
- classes that contain the data are defined and documented;
- relationships between classes are formally specified in an object model;
- criteria that will enable the testing of a software application intended to support the business process are established.

There are six sub-processes involved in the preparation of a project model specification leading to its completion and acceptance for integration within the IFC Object Model:

A21. Break down processes
A22. Define terms and references
A23. Add usage scenario
A24. Analyze model requirements
A25. Develop project object model
A26. Develop test criteria

6.2.2.1 Process A21 - Break Down Processes

Inputs	
Accepted Proposal	*The documentation forming the accepted proposal provides the basis for continuing work. Throughout the specification of requirements, further information will be added to this documentation to provide the basis for the IFC Release Documentation and the IFC Object Model.*
Outputs	
Incomplete Breakdown	*If it is not possible to break down processes to an acceptable degree, the project may be terminated.*
Stage 2 Specification	*Completion of the process breakdown marks a milestone in specification development. It must contain both the high level process diagrams and descriptions of the proposal and further detail of broken down (or decomposed) processes including both additional process models and task descriptions for each activity identified in the process model.*
Constraints	
Peer Review	*Development of the Stage 2 specification through the decomposition of processes should be subject to peer review. That is, domain specialists should validate the work. These may be project partners and others. Wide review of technical development is encouraged for specification development, particularly at this stage. Experts external to the project may be called on for this purpose. A technique that has been used to solicit peer review is publication of process diagrams and description on the World Wide Web*
Performers	
Project Team	*Break down processes to give a finer granularity to the process model. This will allow better identification of the data elements that will form the basis of the object model for the project.*
ITM	*Consideration of the completed stage 2 specification to ensure that all milestone requirements have been met.*

Example:

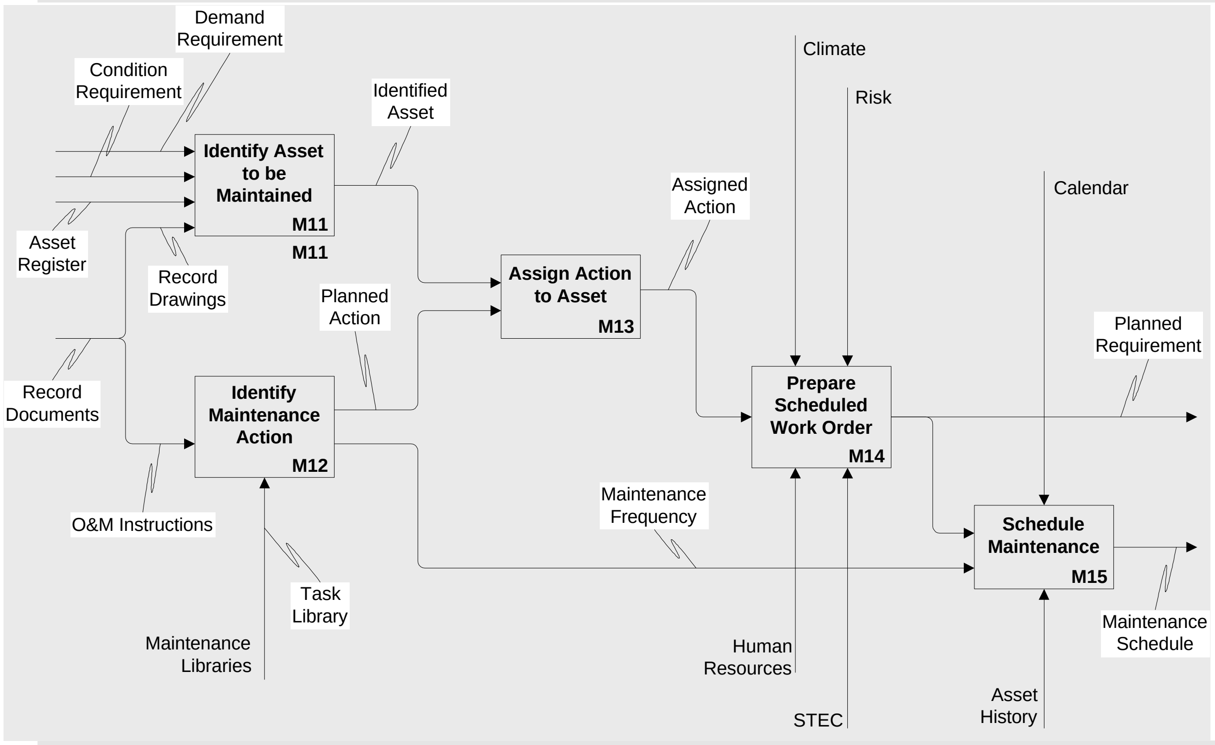


Figure 13: Partially Decomposed Process Model for Maintenance

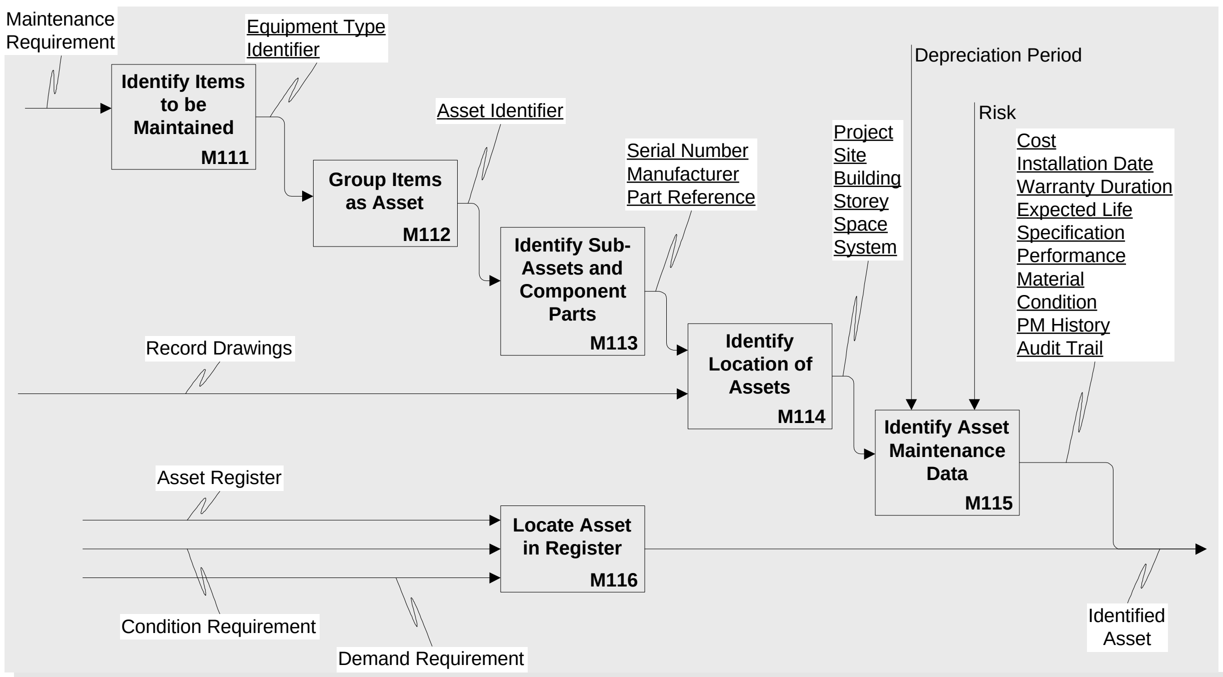

Maintenance Requirement
Equipment Type Identifier
Identify Items to be Maintained M111
Group Items as Asset M112
Asset Identifier
Serial Number Manufacturer Part Reference
Identify Sub-Assets and Component Parts M113
Project Site Building Storey Space System
Identify Location of Assets M114
Depreciation Period
Risk
Cost Installation Date Warranty Duration Expected Life Specification Performance Material Condition PM History Audit Trail
Identify Asset Maintenance Data M115
Record Drawings
Asset Register
Locate Asset in Register M116
Condition Requirement
Demand Requirement
Identified Asset

Figure 14: Fully Decomposed Process Model for Maintenance

Example:

Decomposed Process Task Descriptions

Activity M11 - Identify Assets To Be Maintained

Task Description:

The initial task is to identify objects that are subject to maintenance. This information is obtained from the asset register that identifies the equipment and provides its unique identifier.

Activity M112 - Group Items as Assets

Task Description:

This task involves the specification of the asset. An asset can be either a single object or may be a group of objects upon which a single maintenance action is to be specified. For instance, the following scenarios can be envisaged for asset identification:-

- A large item of equipment such as an air handling unit, boiler or transformer will probably be uniquely identified as an asset.
- An asset may be declared as a group of objects with a spatial identity such as the light fittings within a room where a maintenance action might be that all fittings are to have lamp replacements made at the same time.
- An asset may be declared as a group of objects with a system identity such as the steam traps within a system (or subsystem) where a maintenance action can only be carried out when the system is closed down.

6.2.2.2 Process A22 - Define Terms and References

A vital part of developing IFC specifications is ensuring that the terms used are consistent both within a specification and between specifications. Obtaining agreement about terms used and what they mean (their semantics) is a fundamental part of defining the IFC Object Model.

The importance of this process should not be underestimated, neither should the effort required in achieving consensus about the meaning of terms be undervalued. Getting agreement about what things mean can be one of the hardest tasks in specification development.

Inputs	
Stage 2 Specification	*See A21 above*
Outputs	
Definitions	*Identified documents that are contributing sources of knowledge in the specification development.* *Definitions of terms used in the specification document and particularly as used in task descriptions and usage scenarios.* *Note that the definitions declared at this stage may later also be used to describe the semantics of classes, attributes and properties. However, at this stage, they should relate to the processes described.* *Definitions form part of the stage 3 specification.*
Constraints	
Peer Review	*See A21 above*
Performers	
Project Team	*Preparation and validation of definitions of terms.*
ITM	*Consideration of the completed stage 3 specification to ensure that all milestone requirements have been met.*
Dictionaries etc.	*The dictionaries, glossaries of terms and other documents from which the definitions of terms are derived. The documents used should be declared as Reference Documents in the specification so that others carrying out peer review can consult them..*
Reference Docs	*The documents that provide sources of knowledge to the specification development.*

Example:

Definitions:

Attribute / Relation	**Definition**
AcceptableConditionCriterion	A parameter used to determine whether the condition of an asset is such that maintenance is NOT required or which the condition of an asset must equal or exceed after maintenance.
Access	Constraints existing on access to an asset. **NOTE**: Various types of constraints may exist including access only at certain times; access only if accompanied etc.
ApprovedBy	The person responsible for approving the completed work in accordance with the adopted quality policy.
Asset	A uniquely identifiable element which has a financial value and against which maintenance actions are recorded. NOTE: An asset may be either an individual element or a group of elements whose value is to be considered singly and which is to be the subject of individual maintenance action.
AssetHistory	A record of work carried out on an asset over its life cycle. It records all work orders generated for an asset together with changes in status of the work order until it is completed.
AssetIdentifier	A unique identification assigned to an asset that enables its differentiation from other assets. NOTE: The asset identifier, also frequently termed the asset name, is a type of logical identifier.
AssetRegister	A record of all assets. **NOTE**: An asset register is a table which records the assets of an organization, building or other grouping mechanism and within which the values of all assets can be determined.
Assignee	The person or persons to whom execution of the work order is assigned.
Assignor	The person assigning the work order
………	……

6.2.2.3 Process A23 - Add Usage Scenario

Usage scenarios are developed from the knowledge of industry experts. There are different ways in which to elicit and set down this knowledge:

1. The industry expert sets down his/her own knowledge. This method requires significant external review, as it is normal for an industry expert working in this way to include his/her own assumptions and shortcuts. The review process allows questioning and refinement of the usage scenario through the progressive reduction of implied knowledge.
2. Several domain experts working together can often produce a usage scenario more quickly and effectively. They can continually question and make explicit the knowledge being contributed. A review process is still required to gain consensus.
3. The usage scenario can be developed by collaboration between a knowledge engineer (KE) and one or more industry experts. The KE can make use of structured questioning, interviews and other methods to elicit the required input to the usage scenario. A review process is still required to gain the widest possible consensus.

6.2.2.3. Assertions

An interim procedure that may be adopted is the development of the usage scenario as a set of assertions. This has value in developing the list of classes and their relationships that can then be further amplified during the development of tables and data sheets.

A set of assertions breaks down the usage scenario into simple sentences. Each assertion is the simplest sentence that can be developed. It contains two nouns and a verb, plus some qualifying grammar that is used to describe cardinality and rules.

The objective of developing assertions is to:

- identify classes;
- identify relationships;
- identify cardinality of relationships;
- identify direction of relationships;
- identify attributes;
- identify rules (uniqueness, derived values, constraints).

Example:

	Source	*Relation*	*Cardinality*	*Target*
A	**RectangularGrid**	<u>is a</u>	type of	**Grid**
A	**Grid**	<u>has</u>	a	*Purpose*
A	**BuildingComplex**	<u>contains</u>	zero, one or more	**RectangularGrids**.
A	**Building**	<u>contains</u>	zero, one or more	**RectangularGrids**.
A	**BuildingSection**	<u>contains</u>	zero, one or more	**RectangularGrids**.
A	**Site**	<u>contains</u>	zero, one or more	**RectangularGrids**.
A	**Gridline**	<u>intersects at</u>	one or more	**IntersectionPoints**
An	**IntersectionPoint**	<u>intersection of</u>	exactly two	**Gridlines**
An	**IntersectionPoint**	<u>has</u>	a	*Label*
RULE(derived by concatenating the labels of the intersecting Gridlines).				

Asserted statements such as the above can be valuable as a means of identifying what is a class, what is an attrbute, the name of reationship and the cardinality or numerical constraint of the relationship. This is shown in the above, where classes **classes** are shown in bold, *attributes* are in italic, <u>relationships</u> are underlined and cardinality is plain text..

Inputs	
Stage 2 Specification	*See A21 above*
Outputs	
Usage Scenario	*The usage scenario is a textual description that sets down the complete information requirement in such a manner that it relates together the scope definition, the process model and the object specification.* *At this stage, the specification does not exist. The usage scenario identifies its elements.* *When creating the usage scenario, observe the following rules:* ***<u>Be assertive</u>.*** *A usage scenario should provide a set of assertions that can be modeled and implemented e.g. A building is located on a site.* ***<u>Be clear</u>*** *about the ideas being captured by the usage scenario. The usage scenario provides the basic input to the identification of classes, attributes, relationships, properties and interfaces.* ***<u>Be specific</u>*** *about relationships.* ***<u>Be precise</u>*** *concerning numerical constraints that exist. Frequently, this can be directly understood from the statement in the usage scenario. For instance, in the above assertion, the use of 'a' for building and site implies that one building has a relationship with one site. Be precise also about numerical constraints that exist in the opposite direction. For instance, reversing the roles in the above sentence identifies that 'a site is the location for zero, one or more buildings' implying that one site can have a relationship with many buildings.* *The usage scenario should also identify required information that can be obtained from other domains, e.g. wall dimensions are normally obtained from the Architect and information that is provided to other domains.* *Drawings and sketches improve the quality and completeness of the usage scenario. They also help modelers and software implementers to understand how to realize the classes and relationships that result from the specification. Ensure that drawings and sketches are annotated so that they can be cross-referenced to the ideas expressed in the text.* *The usage scenario forms part of the stage 3 specification.*
Incomplete scenario	*If it is not possible to define usage scenarios to an acceptable degree, the project may be terminated.*
Constraints	
Peer Review	*See A21 above*
Performers	
Project Team	*Development of the usage scenario. Wherever possible, the usage scenario should be based on a real world situation. It is probable that some simulation of usage will be required even in this case since it is likely that a given real world situation may not cover all of the business needs for which the project is developing the model.* *However, be cautious. An existing real world scenario will represent a situation 'as-is'. The process and object models being developed may represent a business improvement by describing a 'to-be' scenario. In this case, simulation will almost certainly be required.*
ITM	*Consideration of the completed stage 3 specification to ensure that all milestone requirements have been met.*

Example:

Activity M112 - Group Items as Assets

Task Description:

This task involves the specification of the asset. An asset can be either a single object or may be a group of objects upon which a single maintenance action is to be specified. For instance, the following scenarios can be envisaged for asset identification:-

- A large item of equipment such as an air handling unit, boiler or transformer will probably be uniquely identified as an asset.
- An asset may be declared as a group of objects with a spatial identity such as the light fittings within a room where a maintenance action might be that all fittings are to have lamp replacements made at the same time.
- An asset may be declared as a group of objects with a system identity such as the steam traps within a system (or subsystem) where a maintenance action can only be carried out when the system is closed down.

Example Usage Scenario:

At this stage, assets which are to be included within the asset register as individual entries need to be grouped. Within the scenario, the air handling unit is an example of an individual item which will be treated as an asset as also is the boiler.

Assets may also be groupings of individual elements that are treated as a single element for maintenance purposes. Two types of grouping can be identified.

The first is an 'assembly' where several items are physically associated. Physical association means that when the group is complete, it can be physically picked up as a single element. An example of a physical association in the illustration is the pump set which is brings together the two pumps and four valves shown (it may also include other elements such as strainers).

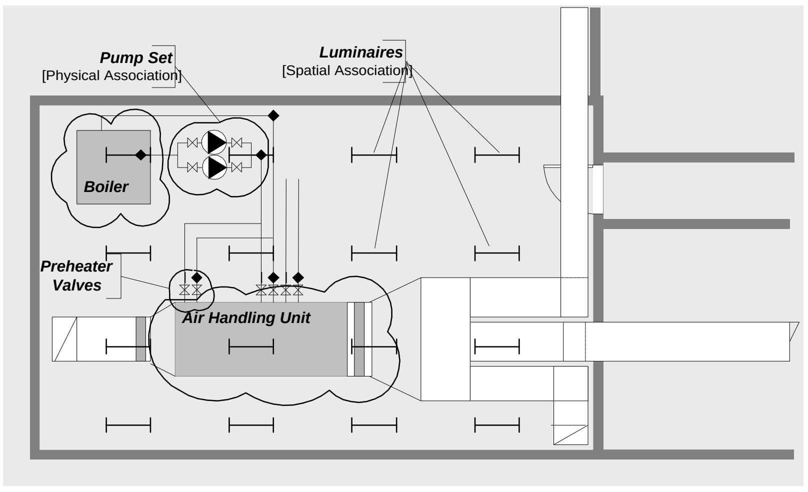


Figure 15: Usage Scenario Diagram **.**

The second type of grouping is a 'group' where several items are spatially associated. Spatial association means that the individual elements within the group remain distinct but that they are all treated as though they were a single element for maintenance purposes. An example of a spatial association in the illustration is the luminaires, each of which is physically distinct but which act together to illuminate the plant room space and are maintained together e.g. for relamping.

6.2.2.4 Process A24 - Analyze Model Requirements

Analyzing model requirements involves considering the information content of the project specification developed to date and isolating the data implied by the various inputs, outputs, constraints and process performers.

Model analysis also identifies classes and data types that meet the needs of the defined processes. Identified classes and data types may already exist within the IFC Object Model (in which case their value is extended to supporting a new process) or they may need to be added in the target Release.

Finally, analysis provides an indication of the information that the current project specification will provide to applications that are compliant with other process specifications and the information that it will receive from such applications.

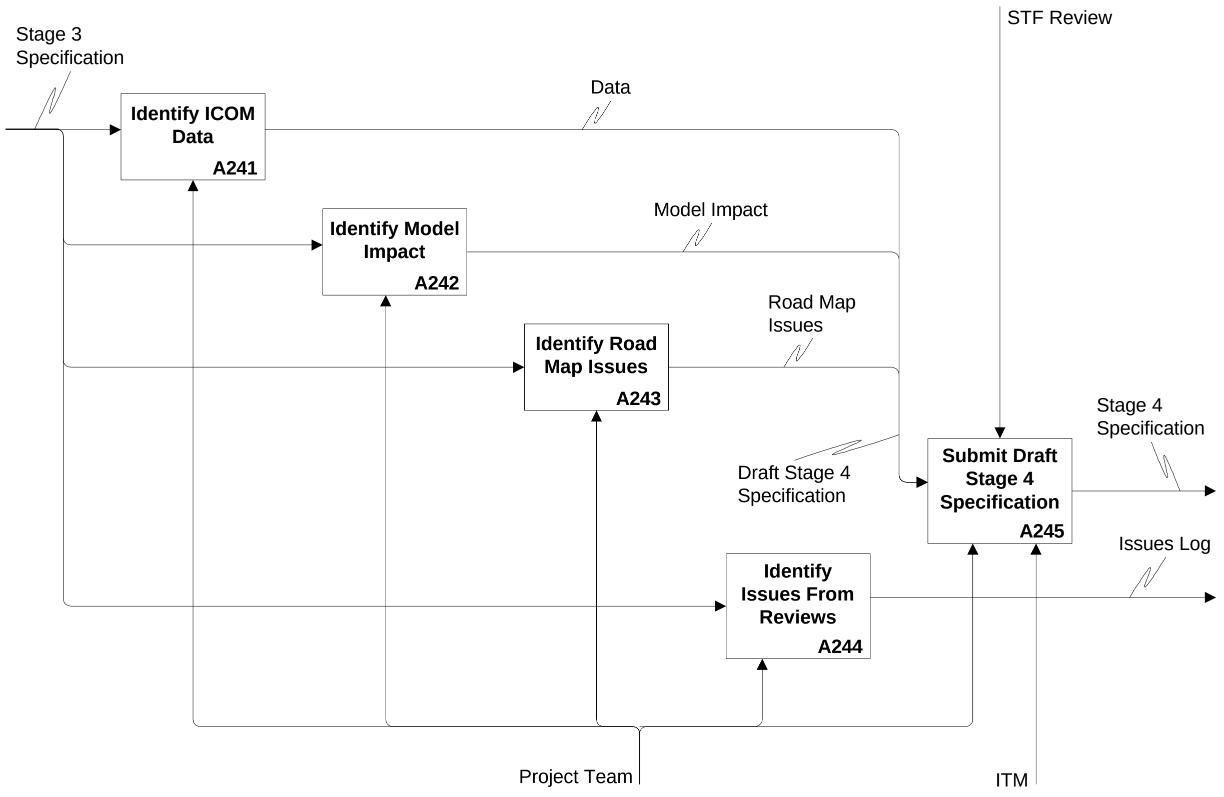

Stage 3 Specification
STF Review
Identify ICOM Data A241
Data
Identify Model Impact A242
Model Impact
Identify Road Map Issues A243
Road Map Issues
Draft Stage 4 Specification
Submit Draft Stage 4 Specification A245
Stage 4 Specification
Issues Log
Identify Issues From Reviews A244
Project Team
ITM

Figure 16: A24 Analyze Model Requirements

6.2.2.4. Process A241 - Identify ICOM Data

ICOM Data are the date elements that can be identified from the process model and that provide the basis for the definition of classes within the object model. ICOM is an acronym that means Input, Output, Constraint and Mechanism (Performer).

Inputs	
Stage 3 Specification	*See A24 above*
Outputs	
Data	*It should be possible to extract information that will be used to determine classes and attributes from the process definitions. If the process models have been fully developed, this will also be visible as input (I), constraint (C), output (O) or mechanism/performer (M) information.* *It is recommended that process models be developed as fully as possible as this provides the easiest means for identifying ICOM data. The arrows to and from the process box directly match the data analysis requirements.* *If the process models are not fully developed, data elements may need to be interpreted from the data descriptions given. For instance, a description such as 'site information' might need to interpreted to provide data elements such as position, address, site area, site perimeter and other elements.* *Data forms part of the draft Stage 4 specification.*
Constraints	
Performers	
Project Team	*Identify data that is exhibited within the process model or that can be interpreted from the process model.*

Example:

Activity M115 - Identify Asset Maintenance Data

Input Information:

- Project
- Site
- Building
- Storey
- Space
- System

Process Performer:

- Depreciation period
- Risk

Output Information:

- Cost
- Installation Date
- Warranty Duration
- Expected Life
- Specification
- Performance
- Material
- Condition
- PM History
- Audit Trail

6.2.2.4. Process A242 - Identify Model Impact

To identify model impact, both classes and data types need to be recognized. In this sense, a class is multi-valued, having attributes that may themselves be other classes or individual data elements whilst a data type has a single value element that is a base type (real number, integer, string, boolean true/false value or an enumerated list)

Inputs	
Stage 3 Specification	*See A24 above*
Outputs	
Model Impact	*This is the point at which the effort is made to relate the specification as it has developed so far to an object model. The model impact considers how the business requirements can be met from existing specifications within the IFC Object Model and what new classes and data types might need to be added to the model.* *Additionally, a project might be able to make use of existing classes but need to add additional attributes. Where possible, a project should be able to identify both new classes and existing classes that need to be extended or modified.* *To carry out this activity effectively, a knowledge of the content of the current IFC Object Model needs to be available to the project. In order to understand what existing classes may be used, it is necessary to understand how classes already within the model are intended to be used.* *Model impact forms part of the draft Stage 4 specification.*
Constraints	
Performers	
Project Team	*Identify new classes and data types needed by the project.* *Identify existing classes within the IFC Object Model that need to be extended or modified for the purposes of the project*

Example:

<u>IFC Model Impact</u>

Usage/Extensions to R2.0 and earlier object types

♦ *None*

New object types

<u>Data Types</u>

♦ *IfcCriterionResult*
♦ *IfcWorkOrderPriorityEnum*
♦ *IfcWorkOrderStatus*
♦ *IfcCriterionValueSelect*

<u>Classes</u>

♦ *IfcAsset*
♦ *IfcAssetHistory*
♦ *IfcAssetRegister*
♦ *IfcCalendar*
♦ *IfcCondition*
♦ *IfcConditionCriterion*
♦ *…..etc*

6.2.2.4. Process A243 - Identify Road Map Issues

Inputs	
Stage 3 Specification	*See A24 above*
Outputs	
Road Map Issues	*Road map issues identify applications that are expected to provide information to an application that is compliant with a model specified by the project and applications that expect information to come from an application that is compliant with a model specified by the project.* *Additionally, road map issues can be used to identify business areas that might be able to provide or benefit from information but which are not yet included as projects in IFC Release cycle planning.* *Road map issues form part of the draft Stage 4 specification.*
Constraints	
Performers	
Project Team	*Identify other models that are related by data to the project model.* *Identify potential models that do not yet exist or are not yet planned that may be related by data to the project model.*
Reference Documents	*Reference documents can be a valuable source of information to assist identification of other industry sectors that might benefit from information provided by the model or which can give information to the model.*

Example:

Road Map Issues

Interoperability Issues

Applications from which information is needed:

- *Building Services*
  - *R1.0: BS-1 HVAC Systems*
  - *R3.0: BS-4 HVAC Load Calculation*
  - *R3.0: BS-5 HVAC System Extensions*
  - *R3.0: BS-6 Performance Metrics*
  - *R3.0: BS-7 Performance Validation*
- *Libraries*
  - *R3.0: XM1: Libraries*

Applications for which information is produced:

- *R1.0: BS-1 HVAC Systems*
- *R3.0: CB1: Client Briefing*

6.2.2.4. Process A244 - Identify Issues From Reviews

Inputs	
Stage 3 Specification	*See A24 above*
Outputs	
Issues Log	*See Issues Resolution Database SECTION 6.2.2.4.4.1 below)*
Constraints	
Performers	
Project Team	*Analyze reviews, create and maintain issues log for the project model.*

Review

It is important that IFC developments are reviewed by industry experts and information modelers other than those who created the IFC specification for the following reasons:

- ensures that the specification is validated by the agreement of a number of industry experts;
- identifies aspects of the domain process which have not been fully included;
- ensures that the specification is internationally applicable;
- provides a wider range of expertise for the development of the specification;
- allows for the correction of inaccuracies within the specification.

The following table identifies typical reviewers, what they should review and their purpose in doing so.

You are …	*You should review …*	*To determine …*
A manager (either within AEC/FM or IT applications)	The project scope statement and usage scenario	Whether this specification meets your business needs
An industry expert, or a modeller responsible for developing specifications and systems that support business processes	All of the above, plus The specification requirements and their supporting material	Whether all the data that you use for the business process is completely and correctly identified
An applications software developer, or a modeller involved in designing and developing systems	The complete project specification and its supporting material	How the specification can be mapped to your application's data, or to the data held in your database(s)

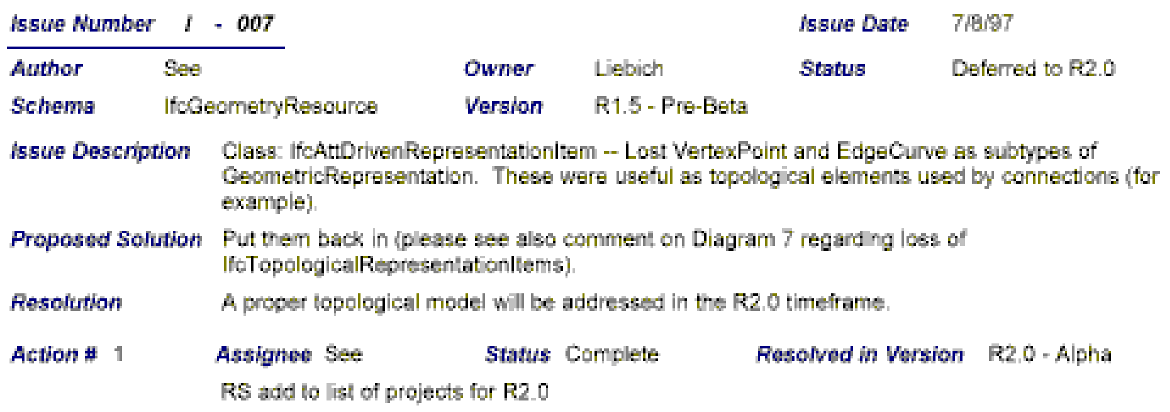
Issues Resolution Datab ase

All issues should be recorded in the Issues Resolution Database for the project with the following items recorded for each issue:

Issue number:	The sequence number assigned to the issue within the Issues Resolution Database for the project
Issue date:	Date on which the issues was raised
Version number:	Version number of the specification against which the issue is raised
Author:	Person responsible for raising the issue, and who may be contacted for clarification of the issue, if necessary
Owner	The person to whom resolution of the issue is assigned
Status: †	Records the status of the issue during its resolution. Initially, all issues have the status "received". Other possible status values are: ***Open***: the issue is still in discussion within the specification development team ***Accepted***: the specification development team has agreed with the issue and identified a resolution,. but has not yet implemented it. ***Rejected***: the issue is erroneous (e.g., refers to an incorrect version of the specification) ***Deferred***: resolution of the issue is deferred to a future IFC release. ***Incomplete***: Resolution of the issue has not yet been completed. ***Resolved***: an agreed resolution has been implemented. On completion all issues should have the status "rejected", "deferred" or "resolved"
Issue Description:	A description of the problem identified.
Proposed Solution:	Proposal, by the author of the issue or the review team, for changes, additions or deletions to the specification or its documentation that will resolve the issue
Resolution: †	Details of the changes made to the specification or its documentation in resolving the issue
Action#:	An issue may have several actions against it, Each action is given a number and has assignee, status and version resolution asociated with it (as below)
Assignee	The person to whom the action is assigned.
Status	The current status of the action
Resolved in Version	The version in which the action is resolved.
Action	The action requiring resolution

(Items marked † are intended for use by the project team, not by reviewers.)

Example:



Issue Number ***I - 007*** ***Issue Date*** 7/8/97

Author See ***Owner*** Liebich ***Status*** Deferred to R2.0

Schema IfcGeometryResource ***Version*** R1.5 - Pre-Beta

Issue Description Class: IfcAttDrivenRepresentationItem -- Lost VertexPoint and EdgeCurve as subtypes of GeometricRepresentation. These were useful as topological elements used by connections (for example).

Proposed Solution Put them back in (please see also comment on Diagram 7 regarding loss of IfcTopologicalRepresentationItems).

Resolution A proper topological model will be addressed in the R2.0 timeframe.

Action # 1 ***Assignee*** See ***Status*** Complete ***Resolved in Version*** R2.0 - Alpha

RS add to list of projects for R2.0

6.2.2.4. Process A245 - Submit Stage 4 Specification

Inputs	
Draft Stage 4 Specification	*The ICOM data, identified model impact and road map issues, when added to the stage 3 specification documents, form the draft stage 4 specification.*
Outputs	
Stage 4 Specification	*The completed project documentation excluding the formal project object model.*
Constraints	
STF Review	*Review of model impact to determine overall scope of the required IFC Object Model integration.*
Performers	
Project Team	*Ensuring that the stage 4 specification are complete and ensuring that all milestone requirements have been met.*
ITM	*Consideration of the completed stage 4 specification to ensure that all milestone requirements have been met.*

6.2.2.5 Process A25 - Develop Project Object Model

This process is concerned with developing the process information and model requirements analysis information into a project object model. This means that classes, attributes, relationships, properties and software interfaces may be defined.

The project object model may be represented in various forms. The key representation is in a structured spreadsheet. This takes the same form as the model representation that is used later by the Specification Task Force during model integration.

Alternatively, a model may be represented in a graphical notation such as the EXPRESS-G notation that is the current default graphical notation used by the IAI. Refer to the EXPRESS-G Readers Guide within this document for further information.

Development of the project object model consists of a set of activities as defined below.

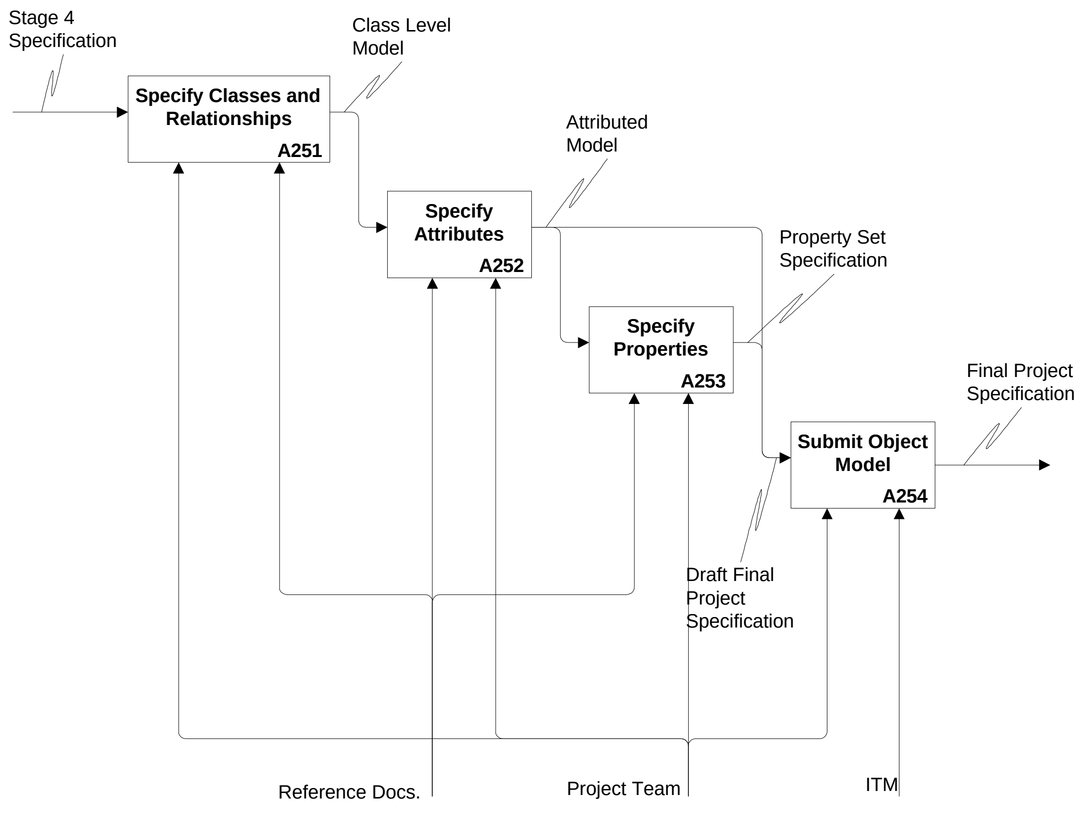


Figure 17: A25 Develop Project Object Model

6.2.2.5. Process A251 - Specify Classes and Relationships

Classes are the fundamental concepts of IFC development. A class may be considered as a container for a range of things that exhibit common definition, properties and behavior. (*An object is an instance of a class which has a unique identity and which has values assigned to attributes so that its state may be determined*).

Inputs	
Stage 4 Specification	*Uses the stage 4 specification document as the primary source from which to build the project object model.*
Outputs	
Class Level Model	*Specifies all of the classes that will be required to support the project objectives and the relationships that exist between the classes.* *Note that class names and relationship names should conform to the naming rules defined within the IFC Object Modeling Guide. Refer to the IFC Object Modeling Guide within this document for further information.*
Constraints	
Performers	
Project Team	*Prepare the class level model.*
Reference Documents	*Reference documents, dictionaries, glossaries etc. used to assist the definition of the meaning (semantics) of the class names.*

Example:

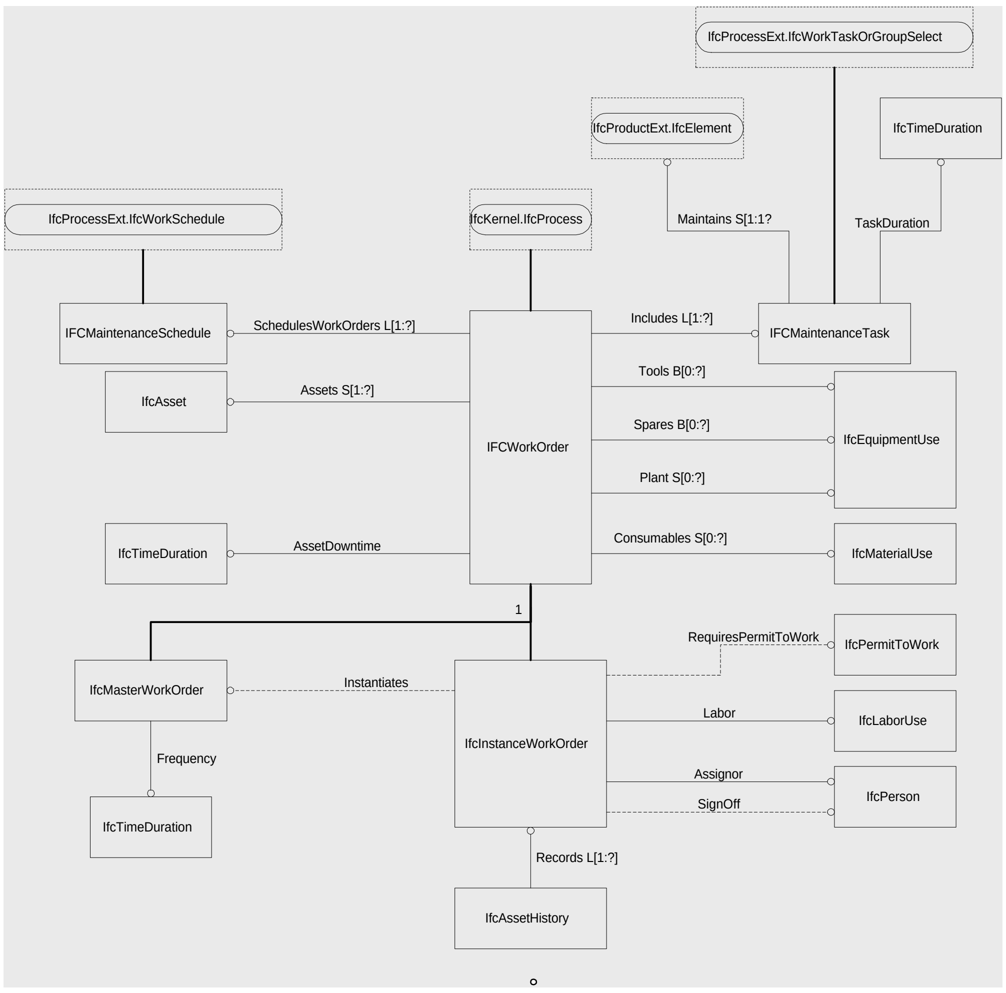


Figure 18: EXPRESS-G Class Level Model

6.2.2.5. Process A252 - Specify Attributes

Attributes define the static properties of a class. When an instance of a class (an object) is used, values are assigned to attributes and these define its state.

Attributes may be either simple data types (real, integer, string, boolean etc.) or derived data types (area, volume etc.) or other classes.

For attributes that are other classes, there is a relationship between the source class (that currently being specified) and the target class (the attribute).

Default values may be applied to an attribute

Where units of measurement for an attribute are appropriate and may be specified unambiguously, they should be identified in the column marked Units. However, it is normal that units may be specified in more than one measurement system (imperial, metric).

Inheritance allows for attributes to be defined within a class at a higher level (parent class) and used within a class at a lower level (child class). For instance, a superclass of Window may be specified to contain a number of attributes. SingleGlazedWindow and DoubleGlazedWindow are both subclasses of Window which have all the attributes of the parent Window class.

Inputs	
Class Level Model	*See sub-process A251 above.*
Outputs	
Attributed Model	*Specifies all of the attributes in the various classes that will be required to support the project objectives.* *Note that attribute names should conform to the naming rules defined within the IFC Object Modeling Guide. Refer to the IFC Object Modeling Guide within this document for further information.*
Constraints	
Performers	
Project Team	*Identify attributes.*
Reference Documents	*Reference documents, dictionaries, glossaries etc. used to assist the definition of the meaning (semantics) of the attribute names.*

Example:

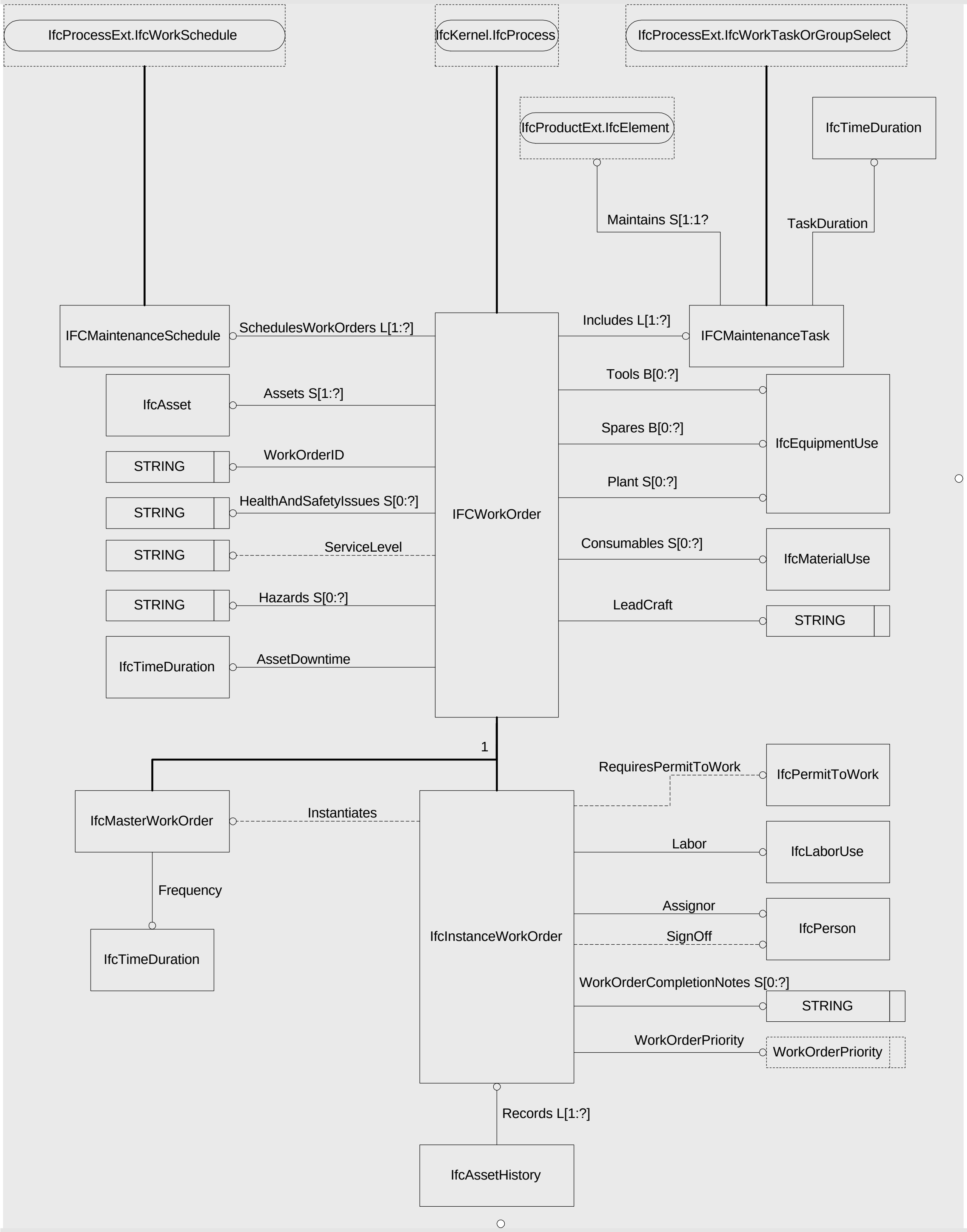


Figure 19: EXPRESS_G Attribute Level Model

6.2.2.5. Process A253 - Specify Properties

Inputs	
Attributed Model	*See sub-process A252 above.*
Outputs	
Property Set Specification	*The specification of all property sets (type driven, non-type driven and extension) that may be used to dynamically extend the IFC Object Model in use at runtime.* *Note that names of properties and property sets and the relationships that exist between property sets should conform to the naming rules defined within the IFC Object Modeling Guide. Refer to the IFC Object Modeling Guide within this document for further information.* *Refer to the Readers Guide to IFC Properties and Property Sets within this document for further information on properties and property sets.*
Constraints	
Performers	
Project Team	*Specify properties and property sets that form part of the object model for the project.*
Reference Documents	*Reference documents, dictionaries, glossaries etc. used to assist the definition of the meaning (semantics) of the property names.*

6.2.2.5. Process A254 - Submit Object Model

Inputs	
Draft Final Project Specification	*The project object model, when added to the stage 4 specification documents, forms the draft final project specification.*
Outputs	
Final Project Specification	*The completed project documentation including the formal object model for the project.*
Constraints	
-	
Performers	
Project Team	*Check to ensure that the project specification is complete and that all milestone requirements have been met.*
ITM	*Consideration of the completed project specification to ensure that all milestone requirements have been met.*

6.2.2.6 Process A26 - Develop Test Criteria

As well as defining the IFC object model specification, the means of testing an implementation of that specification needs to be determined. For this purpose, a part of the development process is the provision of test criteria including the results that should be expected.

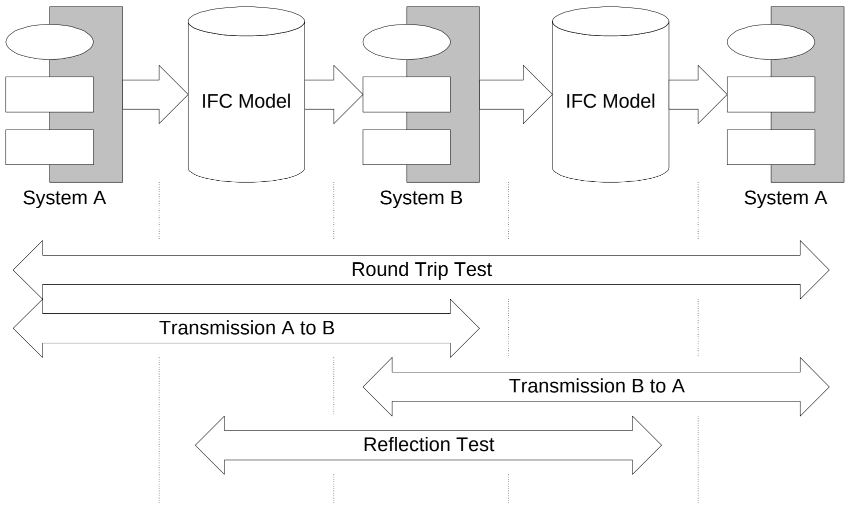


Figure 20: Test Types

Testing is intended to exercise completely the relevant IFC models. Test criteria are defined within the project developing the IFC specification with assistance from the STF and the Software Implementation Committee.

In order to test thoroughly a model, it may be necessary to specify several tests to be carried out in sequence.

Test criteria set down particular circumstances for the use of the specification. Situations which might (or do) exist in the real world are provided with an identification of how classes within the specification are used to achieve particular purposes. In addition, values are given to attributes of the classes so that objects are fully populated with data.

The test criteria are contained within the scope of the project and test only the classes and attributes within the specification that satisfies the project scope.

Three types of test may be used depending on the requirements of the specification:

- Reflection test

  Demonstrates that the implementation under test can write an exchange file and can then read back the file that it originally created. This test will not be carried for implementations of specifications where the information that the software sends and that which it receives are clearly different.
- Transmission A->B

  Demonstrates that an exchange file written by the implementation under test can be correctly read by another proven implementation.
- Transmission B->A

  Demonstrates that an exchange file written by another proven implementation can be correctly read by the implementation under test.

The examples given below are taken from test criteria developed for the cost estimating process defined for IFC Release 1.0

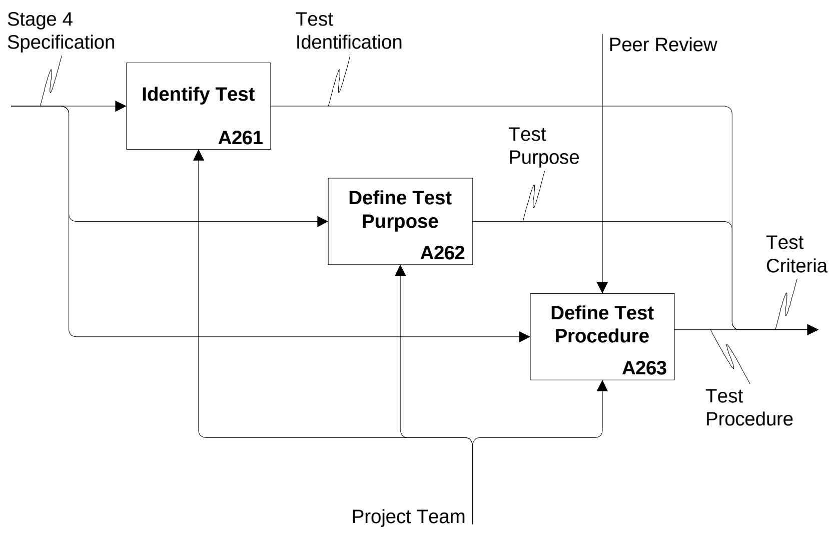


Figure 21: A26 Develop Test Criteria

6.2.2.6. Process A261 - Identify Test

Inputs	
Stage 4 Specification	*See sub-process A245 above*
Outputs	
Test Identification	*Each test should be identified by its position within the sequence of tests and its name.*
Constraints	
Performers	
Project Team	*Name test.*

Example

Test Case 3: Estimating Task & Resource Modeling - Doors

6.2.2.6. Process A262 - Define Test Purpose

Inputs	
Stage 4 Specification	*See sub-process A245 above*
Outputs	
Test Purpose	*The test purpose identifies the reason for carrying out the test and defines the criteria that constitute a successful test. Additionally, the test purpose sets out the real world process that is simulated by the test and how that process relates to other downstream processes reliant upon information provided.*
Constraints	
-	
Performers	
Project Team	*Define test purpose.*

Example

The purpose of this test is to provide for the consistent transfer of door data to and from the IFC model in order to enable modeling of tasks and resources required to construct and install actual objects referenced by the model objects. The test shall ensure that the proposed model completely satisfies the data requirements for task and resource modeling of door installation as implied by the relevant parts of the subject usage scenarios to include the transfer of the Door data.

This test case illustrates a specific usage of task and resource modeling. It uses an estimating system classification and other information in the IfcDoor objects to produce task and resource objects that model door installation for costing purposes. The tasks and resources that are added may be used later by costing and scheduling processes.

6.2.2.6. Process A263 - Define Test Procedure

Inputs	
Stage 4 Specification	*See sub-process A245 above*
Outputs	
Test Procedure	*The test procedure sets out the detail of the test case including:* • *every class used;* • *every object which needs to be instantiated in the test;* • *the value of all attributes within the test;* • *classification of whether attributes are incoming (read from a file) or outgoing (written to a file).*
Constraints	
Peer Review	*See A21 above.*
Performers	
Project Team	*Prepare test procedure by describing what the test is all about, identifying the data values that a conformant application will be tested on and graphically describing the content of the test.*

Example

The first and second floors of the building shall contain 11 interior doors as shown in the drawings. These are 2 types; wooden office doors and double metal staircase doors.

The double metal staircase doors are represented in the model as IfcDoors that contain the following information.

Entity Destription				**Incoming**	
	Attribute Name			**Value**	**comment**
IfcDoor					D-101, D-111
	DoorType			B	
		OperatingType		Double	
	Width			1800 cm	nominal width
	Height			2150 cm	nominal height
	Classification				
		Classifications[1]			
			ClassificationPublisher	Cost System	
			ClassificationTable	Assembly	
			ClassificationNotation	081	
			ClassificationDescription	Door - Hollow Metal	

The wooden office doors are represented in the model as IfcDoors that contain the following information.

Entity Destription				**Incoming**	
	Attribute Name			**Value**	**comment**
IfcDoor					D-102, D-103, D-104, D-105, D-106, D-107, D-108, D-109, D-110
	DoorType			A	
		OperatingType		Single	
	Width			900 cm	nominal width
	Height			2150 cm	nominal height
	Classification				
		Classifications[1]			
			ClassificationPublisher	Cost System	
			ClassificationTable	Assembly	
			ClassificationNotation	082	
			ClassificationDescription	Door - Wood	

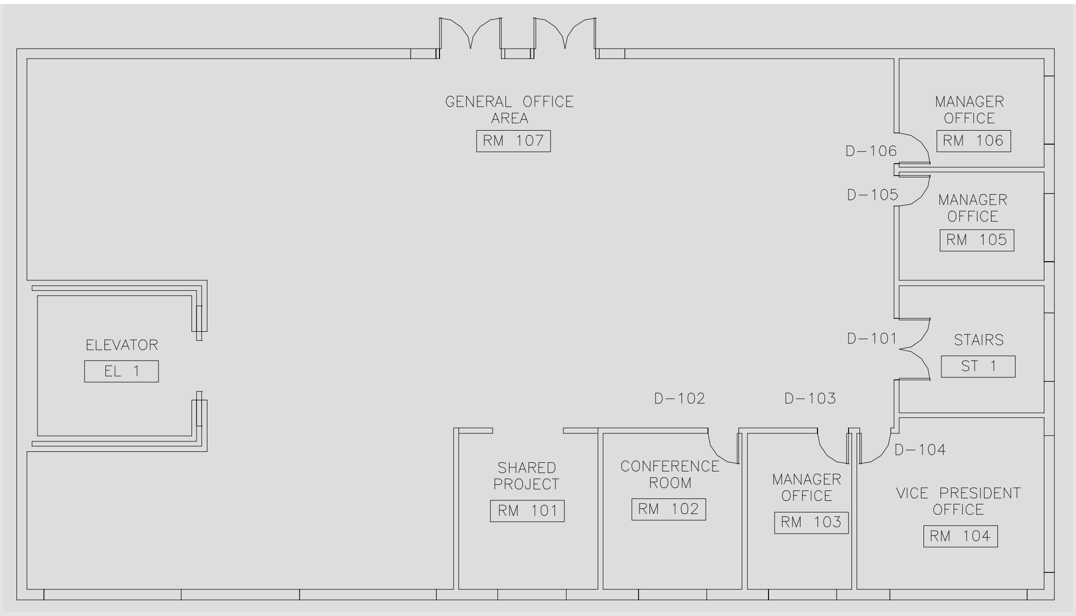


Figure 22: Door layout on floor 1

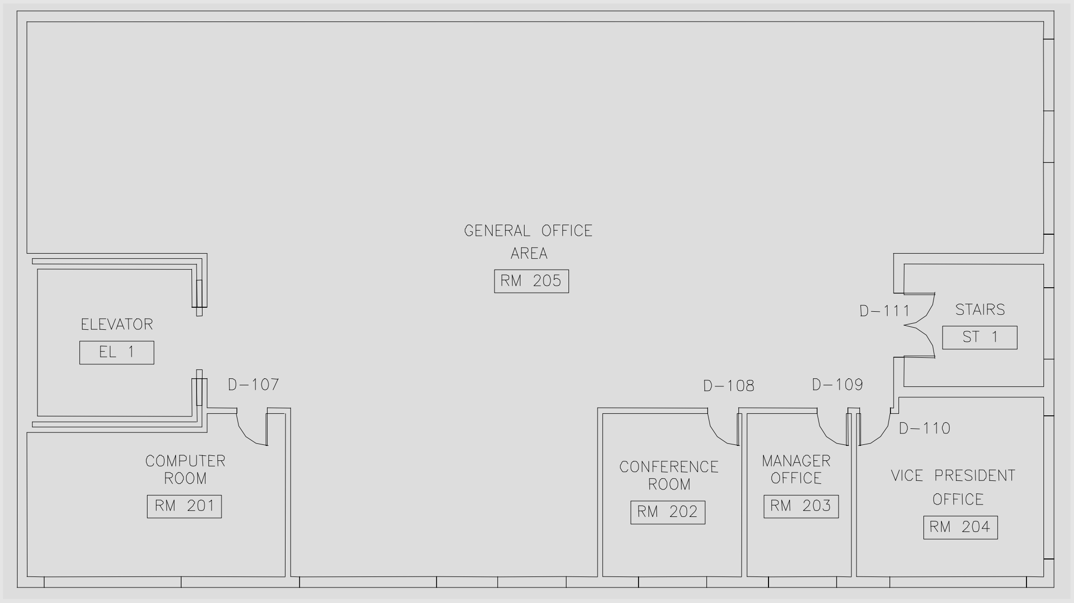


Figure 23: Door layout on floor 2

Using the information in the IfcDoors, the system models the tasks and resources required to install the door. The following charts show the tasks and resources for this test case. In this test case, all materials are modeled as resources. Alternately, some materials may be modeled as IfcManufacturedElements using the HasParts relationship at the IfcProductObject level.

Double metal staircase doors

DoorType				
	WorkSection			
		Tasks		
			Resources	
B				
	Install hollow metal door			
		Frame metal double door frame set		
			Carpenter	
			Metal double door frame set	
		Hang hollow metal door panel		
			Carpenter	
			Hollow metal door panel	
		Install fire door hardware set		
			Carpenter	
			Fire door hardware set	

Wooden office doors

DoorType				
	WorkSection			
		Tasks		
			Resources	
A				
	Install office door			
		Frame office door		
			Carpenter	
			Oak 1x4 framing	
			Oak 1x3 trim	
		Hang office door		
			Carpenter	
			Office door panel & hardware set	

The first step in modeling the installation of one of the doors is to create an IfcWorkSection object and reference it from the IfcDoor using the ResultOf relationship.

Next, for each task required to install the door, an IfcWorkTask must be created.

Finally, for each task, the necessary IfcResourceObjects must be referenced. If the proper resource does not already exist, one must be created. Unlike IfcWorkSection and IfcWorkTask, resources may be shared.

The following chart shows the incoming and outgoing values for the process of modeling the tasks and resources for the stairway exits.

Entity Destription					Incoming	Outgoing	
	Attribute Name				Value	Value	comment
IfcDoor							D-101, D-111
	DoorType				B		
		OperatingType			Double		
	Width				1800 cm		nominal width
	Height				2150 cm		nominal height
	Classification						
		Classifications[1]					
			ClassificationPublisher		Cost System		
			ClassificationTable		Assembly		
			ClassificationNotation		081		
			ClassificationDescription		Door - Hollow Metal		
	ResultOf						IfcWorkSection
		WorkSectionTitle				Install hollow metal door	
		ConsistsOfTasks[1]					IfcWorkTask
			TaskDescription			Frame metal double door	
			Resources[1]				IfcResourceObject
				ResourceType		Labor	
				ResourceDescription		Carpenter	
			Resources[2]				IfcResourceObject
				ResourceType		Material	
				ResourceDescription		Metal double door frame set	
		ConsistsOfTasks[2]					IfcWorkTask
			TaskDescription			Hang hollow metal door panel	
			Resources[1]				IfcResourceObject
				ResourceType		Labor	
				ResourceDescription		Carpenter	
			Resources[2]				IfcResourceObject
				ResourceType		Material	
				ResourceDescription		Hollow metal door panel	
		ConsistsOfTasks[3]					IfcWorkTask
			TaskDescription			Install firedoor hardware set	
			Resources[1]				IfcResourceObject
				ResourceType		Labor	
				ResourceDescription		Carpenter	
			Resources[2]				IfcResourceObject
				ResourceType		Material	
				ResourceDescription		Fire door hardware set	

The following chart shows the incoming and outgoing values for the process of modeling the tasks and resources for the office doors.

Entity Destription						**Incoming**	**Outgoing**	
	Attribute Name					**Value**	**Value**	**comment**
IfcDoor								D-102, D-103, D-104, D-105, D-106, D-107, D-108, D-109, D-110
	DoorType					A		
		OperatingType				Single		
	Width					900 cm		nominal width
	Height					2150 cm		nominal height
	Classification							
		Classifications[1]						
					ClassificationPublisher	Cost System		
					ClassificationTable	Assembly		
					ClassificationNotation	082		
					ClassificationDescription	Door - Wood		
	ResultOf							IfcWorkSection
		WorkSectionTitle					Install office door	
		ConsistsOfTasks[1]						IfcWorkTask
			TaskDescription				Frame office door	
			Resources[1]					IfcResourceObject
				ResourceType			Labor	
				ResourceDescription			Carpenter	
			Resources[2]					IfcResourceObject
				ResourceType			Material	
				ResourceDescription			Oak 1x4 framing	
			Resources[3]					IfcResourceObject
				ResourceType			Material	
				ResourceDescription			Oak 1x3 trim	
		ConsistsOfTasks[2]						IfcWorkTask
			TaskDescription				Hang office door	
			Resources[1]					IfcResourceObject
				ResourceType			Labor	
				ResourceDescription			Carpenter	
			Resources[2]					IfcResourceObject
				ResourceType			Material	
				ResourceDescription			Office door hardware set	

6.2.3. Process A3 - Integrate Model

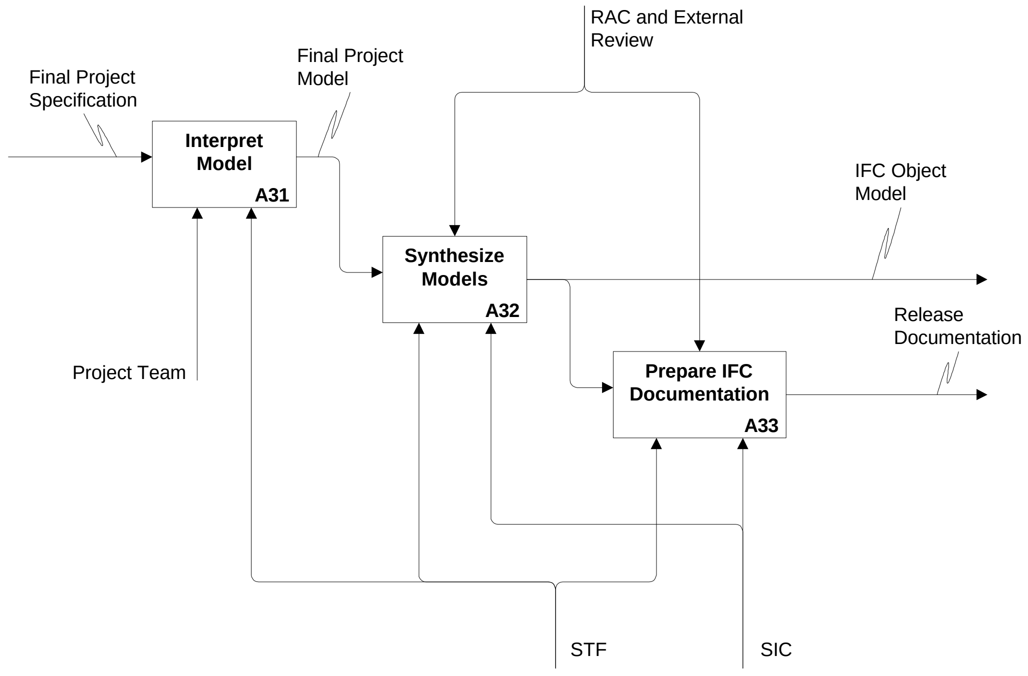


Figure 24: A3 Integrate Model

The objective of model integration is to ensure that:

- the work of all projects is collected into a single, coherent object model that can be implemented;
- information can be exchanged between different software applications using the same model;
- ideas that are the same or similar between different projects are collected together into classes whose meaning and use is not ambiguous;
- documentation to enable detailed understanding of the IFC Object Model is available to implementers.

There are three sub-processes involved in model integration leading to the completion of an IFC Object Model for a given IFC Release:

A31. Break down processes
A32. Define terms and references
A33. Add usage scenario

6.2.3.1 Process A31 Interpret Model

Inputs	
Final Project Specification	*It is necessary to ensure that the final project specification allows consistent understanding by members of the STF who will assist in the model integration process. Consistent understanding requires that the following are available:* • *a usage scenario which can be interpreted commonly by the domain and by the STF;* • *a process model which details the activities of interest;* • *a scope statement which identifies what is in scope for the specification and, equally important, what is out of scope*[4]*;* • *a specification which clearly states* • *the definitions of the objects proposed;* • *their relationships to other objects (including the cardinality or extent of such relationships);* • *constraints which exist on the use of objects or the application of relationships (such as default values, minimum and maximum values, the derivation of values from other attributes or values etc.).*
Outputs	
Final Project Model	*Interpretation comprises a set of assertions, comments and questions raised both by the project team and the STF in an effort to improve the project model and bring it into line with development policies of the IFC Object Model. The source of each assertion, comment or question is identified and each is answered by the project leader who is then responsible for making changes to the project model according to agreements made during interpretation.* *The outcome of this process is a revised model incorporating the agreed changes.*
Constraints	
Performers	
Project Team	*Members of the project team participate in the interpretation process by raising assertions, questions and comments that require response from the project leader.*
STF	*Members of the STF participate in the interpretation process by raising assertions, questions and comments that require response from the project leader.*

Example:

Interpretation Conversation

Domain Team assertions, comments and questions

KM> Feedback by Kirk McGraw on 11 June 1998

KM> Overall, looks very good. I am in favor of more classes and fewer property sets. For example, there are fundamental differences between pipes and ducts.

JF> *Agreed. Many property sets have been promoted in this Alpha Model Draft 3. Refer to the following documentation and attached spreadsheet.*

KM> What is the difference between an attribute that is NotKnown and UnSet? As far as I am concerned there is no difference

JF> *This is a standard convention that we have adopted in IFC's for use with enumerations. An enumeration that is NotKnown is generally one that is currently indeterminate, whereas an enumeration that is UnSet is determinate but not yet defined.*

[4] Items that are considered to be out of scope within one release cycle are often valid ideas for inclusion within a proposal for the next development cycle. Thus, over a period of time, incremental delivery of IFCs can reduce the extent of industrial processes that are out of scope.

STF Team assertions, comments and questions

TL> Feedback by Thomas Liebich on 28 May 1998

TL> General note: 38 new Psets, many of those interconnected by object references, is this appropriate or should some of them be promoted to class level?

JF> Agreed. Many property sets have been promoted in this Alpha Model Draft 3. Refer to the following documentation and attached spreadsheet.

TL> Several Psets concerning Design Criteria: shouldn't Design Criteria be a class in the IFC model, and only its special application to Duct or Duct System be typed?

JF> This is an issue that we need to further discuss. Generally, I agree with this approach, and would suggest that the general constraints mechanism which has come out of the BS-6 project and is proposed for XM-3 should be capable of handling this. We need to arrive at some groundrules for this, and reach agreement on the general constraints model.

TL> Physical Connection sizes: see cross dependencies with IfcPort in network model XM-3

JF> Agreed. This still needs to be done. However, I am uncomfortable with the current definition of IfcPort in XM-3 from the perspective that it mixes and matches both topological and physical elements. I propose that the IfcPort definition in XM-3 be revised to an IfcLogicalPort.

Final Project Model

Modeling Comments	Subtype of		2	3	4	5	6	Attribute / Relation name	Definition
								BS-1 HVAC Duct and Pipe System Design	
New Class Definition promoted from Att_CoordinationRequirement	IfcControl	1						**IfcCoordinationRequirement**	This class captures coordination requirements between different disciplines
								I_CoordinationRequirement	
								OriginatingActor	The actor which originates the coordination requirement
								AffectedActor	The actor which must act upon the coordination requirement
								Requirement	The coordination requirement
The domain team proposed that this class would be called IfcPathwayElement, but its attributes have been moved to the existing R1.5 IfcDistributionElement Class.	IfcBuildingElement	2						**IfcDistributionElement**	This class connects together the physical parts of a networked distribution system.
								I_DistributionElement	
InletPointConnections attribute has been changed to Ports and now points to an IfcDistributionPort.								Ports	References port objects which are used to define the connection ports in the Distribution Element.

6.2.3.2 Process A32 - Synthesize Model

Inputs	
Final Project Model	*At this stage, the responsibility for development of the project model passes from the project team to the STF. It is their role to take this and all other project models for a target Release and synthesize them into the single IFC Object Model.*
Outputs	
IFC Object Model[5]	*See below for information on the Synthesize Model process* *Note that there are several views of the IFC Object Model. These include:* • *Spreadsheet View* *Classes and attributes displayed in a spreadsheet format* • *EXPRESS-G View* *Classes and attributes displayed in the EXPRESS-G graphical notation.* • *EXPRESS View* *A formal specification of the model using the EXPRESS data definition language that is used to create software toolboxes for software implementers.* • *Interface View* *A formal specification of the software interfaces that expose attributes within classes using the IDL interface definition. Software interfaces are limited in extent and capability at this point but are intended to develop to support client/server sharing of information directly between software applications.* *All views of the IFC Object Model are obtained from a single repository that is maintained by the STF on behalf of the IAI.*
Constraints	
Performers	
STF	*See 'What Happens During Synthesis' (section 6.2.3.2.1 below) for detail of the actions taken by STF in synthesizing the model.*

6.2.3.2. What Happens During Synthesis?

Synthesis may have a number of effects on the project model. Each effect is intended to improve the consistency of the model in providing an interoperable solution and to enhance the potential for its implementation.

[5] The approach taken by IAI to model synthesis is similar to the approach taken by the integration process within the ISO STEP development. There are however some differences that have been consciously incorporated by the IAI. The principal difference is in the use if the AIM (Application Interpreted Model) and ARM (Application Reference Model) within STEP as opposed to the single IFC Object Model within IAI. The IFC Object Model is functionally equivalent to the AIM of a STEP Application Protocol. However, whereas STEP has many Application Protocols, IAI has a single model. STEP publishes the ARM as an informative part of its documentation suite, the AIM being the normative part. The project models within IAI are effectively destroyed once their provisions are incorporated into the IFC Object Model (since leaving them in existence would promote the development of non-interoperable applications). The process models of a project are however left intact to demonstrate that the IFC Object Model does satisfy required business needs. In the long term however, it is intended that process models will also be synthesized into a single IFC Reference Process Model as a companion to the IFC Object Model.

Initial work on the IFC Core Model has taken a ‘top down’ approach (that is, seeking to provide a common, generic framework within which specifications can be developed and to provide the necessary bridging structures for interoperability). Synthesis of project models takes a ‘bottom up’ approach (that is, extending the common aspects of the IFC Object Model as these are discovered from work within the project model).

Discovering Generic or Core Classes

Examination of a project model may discover the existence of generic classes that are applicable to every project model. Where such classes are discovered, they will be promoted from the project model to the IFC Core Model.

Discovering Common Classes

Classes that exist at a high level within a project model may express ideas that need to be consistently available to two or more domains or to have a software interface that can be exposed equivalently to several domains.

These classes will be promoted to the interoperability layer so that they are commonly available.

Inheritance from the Core

Project models are intended to use classes provided by the IFC Core Model (kernel, resources and core extensions) either by inheritance or by reference. Technical support specialists should be familiar with classes within the IFC Core Model and inherit from them or refer to them as appropriate[6].

Where an inheritance or reference to a class within the IFC Core is discovered, changes will be made to reflect the fact.

Inheritance from the Interoperability Layer

Project models should inherit from at least one class within the interoperability layer as the first class within the model.

Synthesis will look for relevant classes for a project model to inherit from where these are not already incorporated or will check to ensure that the most relevant class is used as a superclass.

Avoidance of Multiple Inheritance

A fundamental principle of the IFC Technical Architecture is the use of single inheritance. This means that a class can only inherit from one superclass. Synthesis will look for occurrences of a class inheriting from more than one superclass (known as multiple inheritance) and look for alternative ways to achieve the desired effect.

Deep Inheritance

It is good practice to avoid creating deep trees of inheritance (that is, class inherits from several higher levels of superclass). Deep inheritance can cause unnecessary difficulty to software implementers.

Synthesis will look for alternative ways to achieve the desired effect within a specification without the need to use deep inheritance.

Common Specification of Interfaces

[6] Inheritance means that a superclass of that being considered exists within a higher level model and the subclass inherits all attributes, interfaces and behaviors of the superclass. Reference means that the class has a relationship to the referenced class.

Where an interface for a class has been declared, synthesis will look to see if an equivalent interface has already been defined and replace it if necessary. Synthesis will also look to see if other views of a class can be exposed to other domain specifications by the definition of additional interfaces onto a class.

On completion, every class will have at least one interface.

6.2.3.3 Process A33 - Prepare IFC Documentation

Inputs	
IFC Object Model	*See A32 above*
Outputs	
Release Documentation	*Once the IFC Object Model is completed, the documentation that is provided to IAI members and software implementers can be developed. There are various documents that accompany an IFC Release. These are described below..*
Constraints	
Performers	
STF	*Prepare all documentation required for the current IFC Release.*

Documentation that forms part of an IFC Release includes

- An Introduction to the IAI and its IFCs
  The Introduction to the IAI and its IFCs, provides AEC/FM industry professionals with an introduction to the IFC Specifications including the IFC based shared project model concept. It outlines the benefits of IFC compliant applications to end users, provides an overview of IFC, the IAI, and summarizes the processes that have been modeled in this release of the IFC Specifications.

- IFC Model Guide
  The IFC Model Guide provides a reference for the technical requirements, content and arrangement of the IFC Object Model. It includes the following major elements
  - The *IFC Model Architecture* which describes the principles of how the IFC Object Model is organized
  - The *IFC Object Naming and Development Convention* which describes how all of the elements of the model should be named and the guidance rules for the creation of classes and property sets.
  - Samples of parts of the IFC Object Model for information.

  The Guide is intended for specialist object modelers who are interested in IFC development and for software developers who need to understand how the IFC Object Model has been created.

- IFC Specification Development Guide
  The IFC Specification development Guide provides an extended reference on how to develop IFC project specifications in a consistent way. It describes how to develop a project proposal, documentation of processes and classes and development of project object models that can then be synthesized into the overall IFC Object Model. It includes appendices that describe in simple terms some of the technologies used in IFC development. These appendices are useful in learning and understanding why and how IFCs are specified in a manner that is independent of software implementation.

- AEC Processes Supported by IFC
  The AEC Processes Supported by IFC documents the AEC/FM domain processes that the IFC Object Model supports in this release. Therefore, this document effectively defines the scope of AEC project information included in the current IFC Release

- IFC Model Reference

  The IFC Model Reference defines the IFC Object Model. This includes all of the information required by the AEC processes structured in an IFC model detailing object classes, standard interfaces and data types. It also presents several key concepts used in the design of the IFC model including: model structure, capturing design intent, sharing semantic relationships, model extension by application developers, and model exchange versus runtime interface views of the IFC Object Model.

  It also documents the Information Exchange Model view used to represent the IFC Object Model. This Information Model is defined using the international EXPRESS standard and can be used directly by software developers. This document provides an overview of the physical file format.

  Also documented is the runtime interfaces view of the IFC Object Model that complements the Exchange Model view. This view is presented using Object Management Group's Interface Definition Language (IDL), which may also be used directly in software CASE tools to automate parts of the software development process.

  This is the principal document in the IFC Release Document Suite and contains a large quantity of information. It is of value both to end-users and to software implementers. To assist navigation, it is made available primarily in HTML format and can be browsed using Web browsers such as Netscape and Microsoft Internet Explorer.

- IFC Software Implementation Guide

  The IFC Software Implementation Guide provides information and guidance to software programmers on how best to go about developing IFC compliant software. It draws on the experience of those organizations who have already developed such software and contains vital information which can reduce the time (and cost) of development. It discusses potential implementation strategies and includes list of software toolkits and platforms which are available to speed up the development of IFC compliance.

- IFC Implementation Certification Guide

  The IFC Implementation Certification Guide describes the process that has been adopted for certification of IFC compliant software applications and how such compliance must be demonstrated. It also describes the use of the compliance testing toolkit software and how this should be obtained.

6.2.4. Implement and Release

On completion of specification development and when all reviews have been undertaken, software developers will prepare implementations according to the guidance of the Software Implementation Committee.

There is a feedback mechanism existing from implementers to the STF and specification developers so that the results of anomalies found during the course of implementation can be corrected for the formal issue of an IFC Release.

Guidance to software implementers is provided through the Software Implementation Committee of the IAI and is not included in these Guidelines.

7. User Feedback

It is important that users are able to provide comment on the value and applicability of IFCs. This provides a means by which:

- existing parts of the IFC Object Model can be refined and improved to cope with usage situations which may not have been originally considered;
- new processes needing to be supported can be discovered.

Various routes are available for user feedback:

1. Through the software vendor providing the IFC implementation.
2. This is the preferred route since it allows the vendor to determine whether the feedback coming from the user relates to an implementation problem or whether it is genuinely an issue relating to the IFC Object Model.
3. Through the chair of the relevant Domain Committee within the local Chapter.
4. Through the Technical Coordinator of the local Chapter.

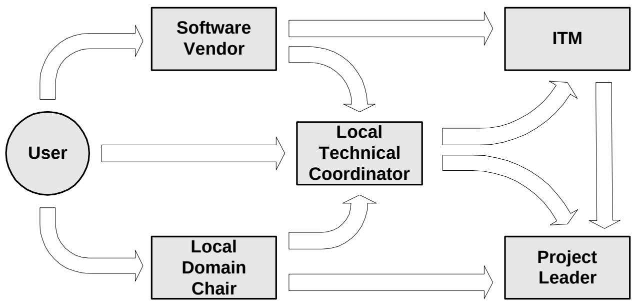


Figure 25 - Routes for Feedback Comments

The comment must be made known to the project leader responsible for the development of that part of the IFC Object Model so that it can be considered and any action required taken. Comments relating to the need for development of additional capabilities are provided to the ITM so that they can consider how and when these should be provided.

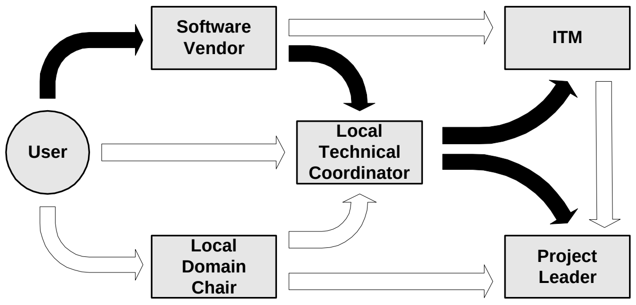


Figure 26 - Comments Routed via Technical Coordinator

8. Providing Feedback

It is anticipated that software vendors will enable users to provide feedback by inclusion of a IFC Feedback form with their software.

When a comment is made, certain things need to be known:

- the IFC release number in use;
- the domain for which the comment is made;

e.g. HVAC

- name(s) of classes/objects which are subject to comment (if these are known or can be determined);

e.g. IfcCoveringElement
IfcElementProfile (new)
StrawFloorCovering (type definition / attribute set)

- identification of attributes needing to be considered;

e.g. ElementProfile list of positions and thickness at each position. Needs to allow for profiling on at least two axes.

- comment;

e.g. Extend model to support the process of HVAC design for Cowsheds.
Provide for varying thickness coverings in HVAC model

- what action caused the comment to be made;

This is probably the most important information and should be given in terms of domain language and NOT in terms of what the software was doing.

e.g. Design of the HVAC systems for a cowshed requires that the thermal characteristics of the straw covering the floor need to be included. The information received from the Architect included IfcCoveringElement objects with a uniform thickness. However, for the purposes of design, it is necessary to allow for varying thickness. The HVAC model did not allow the addition of this varying requirement.

- sketches and diagrams should be included with the comment to help with identifying how it should be best considered.

Appendix A - IFC Object Model Architecture

This section provides an outline of the IFC Object Model Architecture used for development of IFCs. For a complete description, please refer to the document 'IFC Model Architecture'

The principles of the IFC Object Model Architecture are to;

- provide a modular structure to the model;
- provide a framework for sharing information between disciplines within AEC/FM;
- ease the continued maintenance and development of the model;
- enable information modelers to reuse model components;
- enable software authors to reuse software components;
- facilitate the provision of upward compatibility between model versions.

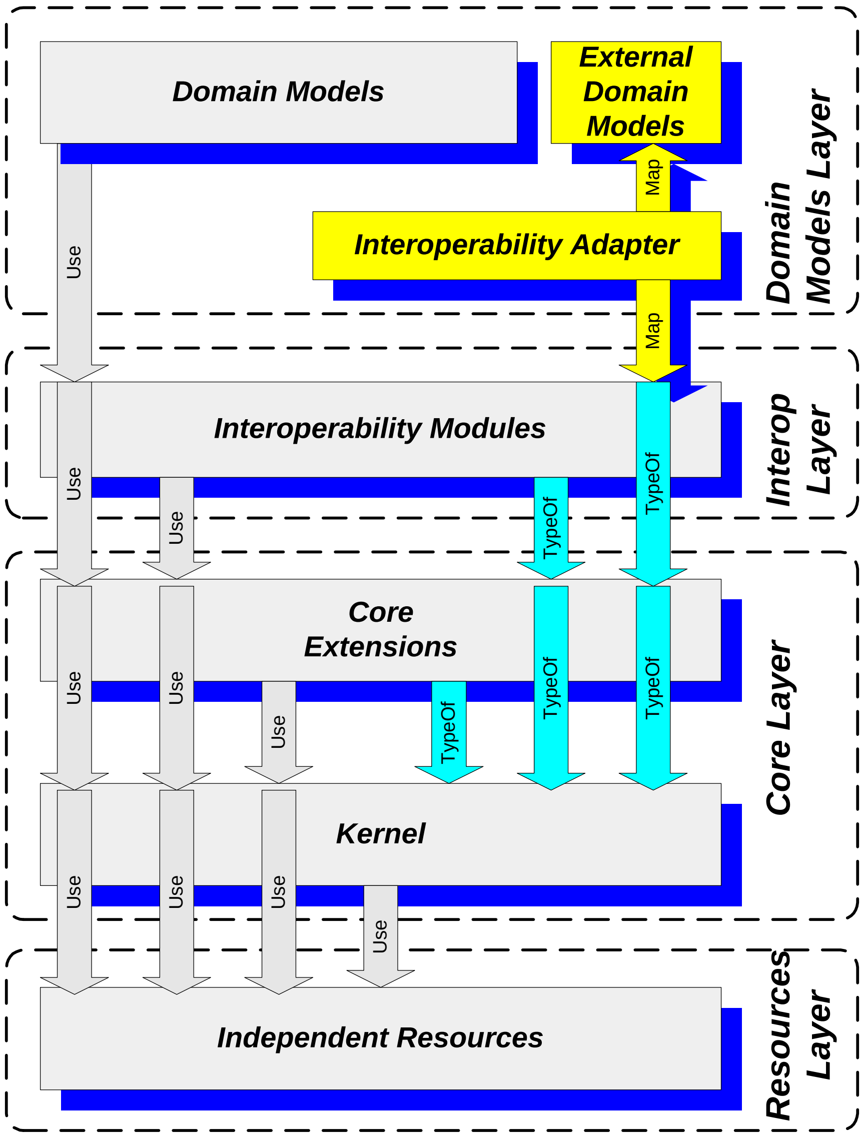


Figure A1. IFC Layered Model Architecture

There are four conceptual layers within the architecture. Within each layer a number of grouped modules exists.

Layer 1 - Independent Resources

Independent Resources are ideas that do not rely on any class described within the Kernel for their existence. For instance, geometry can be created as a resource and then used to describe the shape of an object defined in the Kernel or Domain Extension models.

Independent resources form the lowest layer in IFC Model Architecture and can be used or referenced by all classes in the other layers.

Layer 2 - Core

The Core contains two sub-layers namely Kernel and Core Extensions

Kernel

Provides basic, shared concepts in AEC/FM projects and determines the model structure and decomposition. The Kernel includes fundamental concepts concerning the provision of objects, relationships, type definitions, attributes and roles.

The Kernel provides the bridge between individual domain requirements and the platform for all model extensions. Kernel constructs are a mandatory part of all IFC implementations.

Kernel classes may reference classes in the Independent Resources but may not reference those in other parts of the Core or any classes in the Domain Models.

Core Extensions

A Core Extension is a specialization of classes defined in the Kernel such as Product and Process. Primary relationships and roles are also defined within the Core Extensions.

A class defined within a Core Extension may be used or referenced by Adapters and Domain Models, but not by a class within the Kernel or Independent Resources.

Layer 3 - Interoperability

The interoperability layer provides an interface for one or more Domain Models. It fulfils three requirements:

- it enables plug-in of Domain Models which directly use or reference Core definitions;
- it enables plug-in of externally developed Domain Models using a mapping mechanism down to Core definitions;
- it provides an exchange mechanism above the Core to enable interoperability across domains.

Layer 4 - Domain Models

Domain Models provide further model detail within the scope requirements for an AEC domain process or a type of application. Each is a separate model that may use or reference any class defined in the Core and Independent Resource layers.   Examples of Domain Models are Architecture, HVAC, FM, Structural Engineering etc.

Appendix B - IFC Model Development Guide

Introduction

The purpose of this Model Development Guide is to set down rules, conventions and guidelines to be used in developing the IFC Object Model. The Guide covers all formal views and parts of the model including the use of EXPRESS-G graphical notation, the EXPRESS data definition language, the OMG/CORBA Interface Definition Language (IDL), classes property sets and interfaces.

The format of the Guide is:

- The Guide contains a number of sections, each section setting out the rules, convention or guidelines related to a particular aspect of the IFC Object Model.
- Each rule, convention or guideline is numbered. Numbering is by section followed by the sequence number within that section.
- Each rule, convention or guideline has a name that is intended to provide a terse description of its purpose
- Each rule, convention or guideline may have accompanying text or graphics that provide further explanation on its application.

0.	***NOTATIONS***	
0.1	**Definition of a Model** The term model shall mean a representation in conceptual form of the flow and content of information that is contained within a defined area of interest (termed the Domain of Discourse)	*to stand in place of what is real so that something useful can be determined about how the reality will behave in the real world. As an analogy, consider the use of model aircraft in wind tunnel studies to assess the aerodynamic behavior of the full scale aircraft. Because interest is only centered on airflow over the surface of the aircraft, this part is modeled accurately. The remainder of the full aircraft is not of interest in this context and so is not modeled.* *A model of information enables us to see the flow, structure and development of the information content of AEC/FM over its lifecycle. Within a particular IFC project, only a part of the total information flow and content is of interest. This is modeled as accurately as possible. The remaining information flow and content is not of interest to the project and so is ignored if it does not impact upon the projects area of interest.*
0.2	**IFC Model Requirements** At the current stage of development, models shall describe the flow of information within the defined area of interest (the process model) and the information content within the defined area of interest (the object model)	*The term object model is used to describe information content in preference to other possible terms including information model, data model, product model, project model etc.*

0.3	**Process Model Graphical Notation** The preferred graphical notation of process modeling is the IDEF0 notation in accordance with Federal Information Processing Standard (FIPS) 183.	*IDEF0 is the preferred form of process modelling notation from IFC Release 2.0 onwards. A number of Release 3.0 projects have used the IFC-PDEF notation previously preferred and this is acceptable.* *From Release 2.0 onwards, use of UML Use Cases (including the provision of collaboration diagrams) are also considered to be an acceptable form of process modelling notation*
0.4	**Object Model Graphical Notation** The preferred graphical notation of object modeling is the EXPRESS-G notation in accordance with ISO 10303 part 11.	
0.5	**Object Model Data Definition Notation** The preferred data definition language of object modeling is the EXPRESS language in accordance with ISO 10303 part 11.	
1.	***LANGUAGE***	
1.1	**Use of English** The language used in the development and documentation of the IFC Object Model shall be English.	
1.2	**Spelling** The spelling of all words shall follow the conventions of American English usage. That is, use 'z' in words such as organization, use 'or' in words such as color and labor etc.	
1.3	**Authority** Authorized spelling shall be as denoted in the Microsoft Spelling Checker for English (United States) Office 97 edition.	
2.	***NAMING CONVENTION***	
2.00	***Names Generally***	
2.01	**Case of Names** All names of schema, classes, property sets, relationships, enumerations, select types, defined data types, attributes, functions, rules and interfaces shall be written in upper and lower case characters as a single name without spaces. The first character of each word in normal usage shall be written as an upper case character. All other characters forming part of the same word in normal usage shall be written in lower case characters.	*The convention used is based on that promoted by Microsoft but does not fully follow the Microsoft variable naming convention in that it does not identify the data type within the name.* *Note that the IFC naming convention differs from that used in the ISO10303 (STEP) development in which entities (classes), relationships and data types are named in lower case with complete words separated by an underscore (_) character*
2.02	**Length of Names** There are no restrictions placed on the length of names for schema, classes, property sets, relationships, enumerations, select types, defined data types, attributes, functions, rules and interfaces.	*It is better to ensure that names are clear than that they are short. This also provides semantic clarity. For instance,* **'DesignIntentCumilativeOccupancyNumber'** *indicates attribute meaning clearly whereas abbreviating the name as '***DICON'** *by using the first character of each word is unclear .*
2.1	***Prefix and Suffix Names***	

2.11	**Schema Names** Schema shall be identified as to the layer within the IFC Technical Architecture at which they exist as part of the schema name as follows: • **Resource Layer** Schema name has the term 'Resource' appended • **Core Layer** Schema name has the term 'Extension' appended • **Interoperability Layer** Schema name has the term 'Shared' included after the Ifc prefix and before the name describing the content. • **Domain/Application Layer** Schema name has the term 'Domain' appended	*For example, the following schema identify the use of the layer identification:* • ***Resource Layer*** *IfcGeometryResource schema* • ***Core Layer*** *IfcProcessExtension schema* • ***Interoperability Layer*** *IfcSharedBuildingServices schema* • ***Domain/Application Layer*** *IfcFacilitiesManagement schema*
2.12	**Ifc Prefix** All names of schema, classes, enumerations, select types, defined data types, functions and rules shall be prefixed by the term 'Ifc' to identify their usage within the IFC Object Model. The prefix 'Ifc' shall be treated as a word in normal usage and the 'Case of Names' rule applied to its use. Note that the prefix 'Ifc' shall NOT be added to names of attributes and relationships or to names of property sets (see Pset Prefix rule below).	
2.13	**Enum Suffix** The name of all enumeration data types shall be suffixed with the abbreviated term 'Enum'. The suffix 'Enum' shall be treated as a word in normal usage and the 'Case of Names' rule applied. The only exception is the adapted use of enumerations from STEP (ISO 10303), where the suffix 'Enum' shall not be added to the name given by STEP.	
2.14	**Enum Suffix for Generic Types** If the enumeration is used to denote the generic type of the class, is shall be suffixed with the term "Type", followed by the abbreviated term "Enum". The suffices 'Type' and 'Enum' shall be treated as words in normal usage and the 'Case of Names' rule applied.	
2.15	**Select Suffix** The name of all select data types shall be suffixed with the term 'Select'. The suffix 'Select' shall be treated as a word in normal usage and the 'Case of Names' rule applied. The only exception is the adapted use of enumerations from STEP (ISO 10303), where the suffix 'Select' shall not be added to the name given by STEP.	
2.20	***Classes***	
2.21	**Classes** The name of the class shall be a noun or combination of nouns, denoting the "content" or "type" of the class.	

2.22	**Relationship Classes** All classes acting as objectified relationships classes within the IFC Object Model shall contain the term 'Rel' following the 'Ifc' prefix and before the name of the class in normal usage. The inserted 'Rel' shall be treated as a word in normal usage and the 'Case of Names' rule applied. The name of the objectified relationship class shall be a verb that denotes the "function" of the objectified relationship class.	*In this case, although the primary name of the class is given as a verb and not a noun (apparently contravening rule 2.21), the addition of the Rel prefix is sufficient to allow the class name to act as a noun. For instance, the term 'groups' does not function as a noun whereas the term 'RelGroups' does, thereby satisfying rule 2.21.*
2.23	**Attributes in Relationship Classes** Each objectified relationship class defines two major attributes, the relating (left side) and the related (right side) class, which are put into a relationship by virtue of the objectified relationship class. The following naming convention applies for both attributes. The name is prefixed by either 'Relating' or 'Related' and the name shall be identical with the name of the class to which the attribute relates (without prefix 'Ifc'). In case of a one to many relationship, the name shall be given in plural.	
2.30	***Relationships***	
2.31	**Composition Relationships Prefix** All relationships between classes that express a composition aggregation shall be prefixed with the term 'PartOf' and followed by the name of the class that acts as the aggregate (without prefix 'Ifc'). The 'PartOf' relationship shall be the direct relationship, complemented by the inverse 'Has' relationship. The inverse relationship shall be prefixed with the term 'Has' and followed by the name of the class that acts as the aggregated item (without prefix 'Ifc').	
2.32	**Association Relationship** All relationships between classes that are not aggregations are associations.	
2.40	***Property Sets***	
2.41	**Pset Prefix** All names of property sets shall be prefixed by the term 'Pset_'. The prefix 'Pset_' shall be treated as a word in normal usage and the 'Case of Names' rule applied.	
2.42	**Common Property Set** Common Property Sets (valid for all instances of the typed class) shall be named using the Pset prefix and then concatenating the name of the class (without "Ifc" prefix) and the "Common" suffix.	*For furniture, the common property set Pset_FurnitureCommon could apply.*
2.43	**Generic Property Set** Generic Property Sets (valid for all instances of the typed class having a generic type value) shall be named using the Pset prefix and then concatenating the name of the class (without "Ifc" prefix) and the name of the generic type attribute of the class.	*For furniture, the generic property set Pset_FurnitureType could apply.*
2.50	***Interfaces***	

2.51	**Interface Prefix** All names of interfaces shall be prefixed by the term 'I_', The prefix 'I_' shall be treated as a word in normal usage and the 'Case of Names' rule applied to its use.	
3.	***MODEL SPECIFICATION RULES***	
3.1	**Single Inheritance of Subtypes** A subtype shall be a specialization of exactly one supertype. That is, we are using single inheritence only.	
3.2	**Exclusion Constraint in Supertypes** A supertype shall be constrained so that it can be instantiated exclusively by one of its subtypes. That is, within the EXPRESS language view of the IFC Object Model, only the ONEOF supertype constraint shall be used.	
3.3	**Substitution Principle** The substitution principle of Liskow asserts that: *"It must be possible to substitute any object instances of a subclass for any object instance of a superclass without affecting the semantics of a program written in terms of the superclass."* The substitution principle shall be followed. Redefined types shall not be included within the IFC Object Model.	
3.4	**Mandatory Attributes and Relationships** Attributes and Relationships shall be optional by default (for single attributes and relationships using the OPTIONAL keyword, for aggregates using the bounds [0:?]. Therefore those optional attributes and relationships are not required for instantiation in exchange sets for certification. Only those attributes and relationships with are required for instantiation in all exchange sets shall be made mandatory.	
3.5	**Optional Aggregation** Aggregation relationships (LIST, SET, BAG) that may be specified as empty shall be shown as a mandatory relationship with a cardinality of zero to many and NOT as an optional relationship of one to many. For example, for a LIST relationship, the EXPRESS code shall read LIST [0:?] OF <type or class> and not as OPTIONAL LIST [1:?] OF <type or class>.	
3.6	**Relationships Across Schema Boundaries** Relationships between classes that span schema boundaries shall be handled by the introduction of a Relationship Class.	
3.7	**Many to Many Relationships** All many-to-many relationships shall be resolved to one to many or one to one relationships objectified through the introduction of a Relationship Class.	

4.	***SCHEMA LAYERING***	
4.1	**References between layers in the architecture** The architecture operates on a 'ladder principle'. At any layers, any class may reference or use a class at the same or lower layer but may not reference or use a class from a higher layer. References within the same layer must be designed very carefully. Currently the following layers are defined (starting from low): • Resource layer • Core layer • Interoperability layer • Domain/Appplication layer	
5.	***INTERFACE RULES***	
5.1	**Default Interface** Every class shall have at least one interface that is defined as default. The default interface shall expose all of the attributes within its associated class. The name of the default interface shall be the interface prefix followed by the name of the class whose attributes it exposes (without prefix ‘Ifc’).	*The provision of a default interface that exposes all of the attributes within a class conforms to the provisions of ISO 10303 part 26 which defines an IDL binding for EXPRESS.*
5.2	**Class Interfaces** A class may have additional interfaces that expose a defined set of attributes within that class. Each additional interface shall be named using the I_ prefix (rule 2.51) followed by a name that is indicative of the purpose of the interface.	
5.3	**Overlapping Interfaces** An attribute of a class may be exposed through multiple interfaces.	
5.4	**Interface on Multiple Classes** An interface may expose attributes from more than one class.	
5.5	**Interfaces Within Schema** An interface that exposes attributes from multiple classes shall be restricted to the exposure of attributes from classes that exist within a single schema. An interface shall not cross schema boundaries.	
6.	***PROPERTY SET RULES***	

6.1	**Use of a Property Set Instead of a Classes** Property Sets provide a mechanism for dynamically extending and changing IFC objects. Some object data will remain stable through the lifecycle of an object. Another case is different property sets for different objects of the same type even though at the same stage. Other data will be added and removed throughout the lifecycle. In the latter cases, use of property sets is preferred as these changes can be made at runtime and do not require a class change.	
6.2	**Indications for using Property Sets** The information does not appear in exchange sets outside of a domain (or possibly a "stage" in a domain). The information appears in multiple Property Sets. The information within the Property Sets shall be self contained, i.e. they do not rely on other information provided otherwise in the static part of the IFC model	
6.3	**Indication of not using Property Sets** If there is need to have independent product shape representation for information items handled by Property Sets, this shall be done by promoting this part of the property set to a class and using the IfcProductShape mechanism on the semantic part of the model (this does not apply to –non driving– dimension parameters). If the information within the Property Sets is essential for the class, then it should be specified as explicit attribute on class level. If the information is in multiple exchange sets (*Suggestion)*	
6.4	**Depths of Property Set nesting** Multiple levels of nesting in Psets can be very difficult to understand, control and prove and implement. Therefore, nesting should never be carried beyond three levels of Pset. That is, no more than two levels of reference below the type defining Pset.	

Appendix C Readers Guide to Process Modelling Using IDEF0

Preferred Notation

From the completion of the work by project teams on IFC Release 3.0 projects (approximately March 1999) onwards, the preferred notation for the creation of graphical process models for IFC specification projects is IDEF0. This preference replaces the IFC-PDEF notation preferred for creating process models prior to that date.

The IDEF0 notation (Integration Definition for Function Modeling) was originally developed during the 1970s as part of the U.S. Air Force Program for Integrated Computer Aided Manufacturing (ICAM) and was formalized by publication of the IDEF manuals in the early 1980s. In 1993, the U.S. National Institute of Standards Technology (NIST) documented the notation as Federal Information Processing Standard 183. This document should be consulted as the authoritative reference on the notation.

IDEF0 is a widely used notation for the creation of process models and has been selected as the preferred notation by IAI for the following reasons:

- Formal documentation support
- Extensive software support

Purpose of a Process Model

A process model describes the activities that exist within a business process. A scope statement that sets out, in broad terms, the content of the business process and the process model that exposes it.

The process model defines all of the required activities and sets them into a logical sequence. This sequence is driven by the dependency of one process on the information that is provided to it by one or more other processes. It is NOT time based and should not be confused with scheduling of tasks as may be represented in a GANTT chart or PERT diagram.

A process model can be developed to a very fine or very coarse degree of detail. The more precise the model, the more specific it becomes to a particular process as practised in one place. If it is less precise, it can be used with a high degree of generality.

Process models are used in IFC specification development projects as the means to discover and capture the information content of a business process and how that information is to be exchanged between participants in the process.

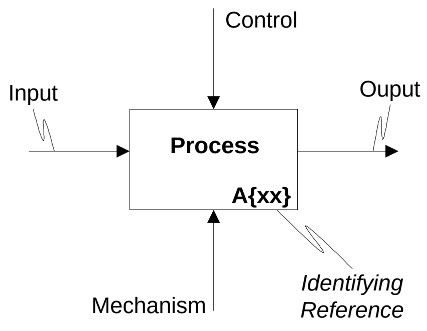
Process models can also be used for other purposes including:

- Quality Assurance

  A quality manual expresses activities to be undertaken, sequences of activities, roles and responsibilities and audit requirements. All of these can be expressed within a process model.
- Business Process Improvement

  A process model enables the capture of ‘as-is’ information about a process. This model can then be analyzed and redeveloped as a ‘to-be’ process model that describes improvements.

The Elements of a Process Model



A process is shown in a process model as a rectangular box. It contains a unique text description or label that describes what the process is. The label may contain several words and these are usually justified about the centre point of the box (both horizontally and vertically). The size of the process box can be increased to enclose the description.

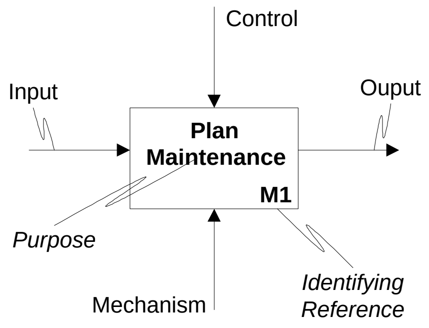
A process is an action. Because of this, the label is expressed as a verb phrase. For instance, in a process model, the maintenance planning process in FM would be labeled as 'Plan Maintenance' and not as 'Maintenance Planning'.

Every process in a process model has a unique identifier shown in the bottom right hand corner.

☞ **Process labels and identifiers are unique within a process model.**

A line entering the process on the left shows INPUT. Each input has a label that describes information used by the process.

A line leaving the process to the right shows OUTPUT. Each output has a label that describes information delivered by the process.



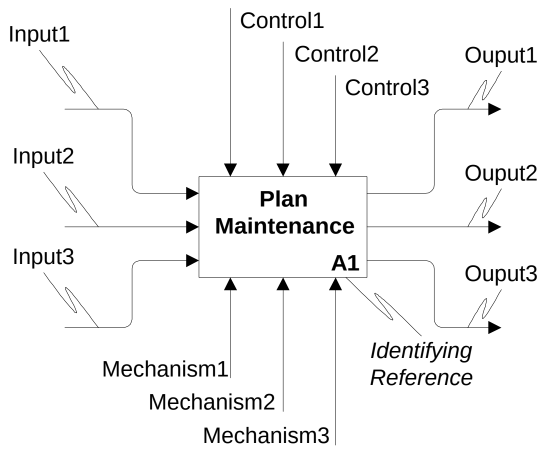
Completion of the process may be subject to one or more controls that constrain the way in which the process may be undertaken.

A line entering the top of the process shows a CONTROL. Each control has a label that describes the constraint on the process.

Completion of the process may use one or more mechanisms that assist or have an involvement with its undertaking. A mechanism is an actor in the process and may be a person, a database or software that is used. Generally, IFC process models show the organizations involved in the performance of the process and may also be referred to as PROCESS PERFORMERS. A line entering the underside of the process shows a mechanism. Each mechanism has a label that describes what it is.

☞ **Inputs enter at the left**	**Controls enter from above**
☞ **Outputs leave from the right**	**Performers enter from below**

Multiple Inputs and Outputs



A process may have several inputs, controls, mechanisms/performers and outputs. Each of these should be shown and labelled individually.

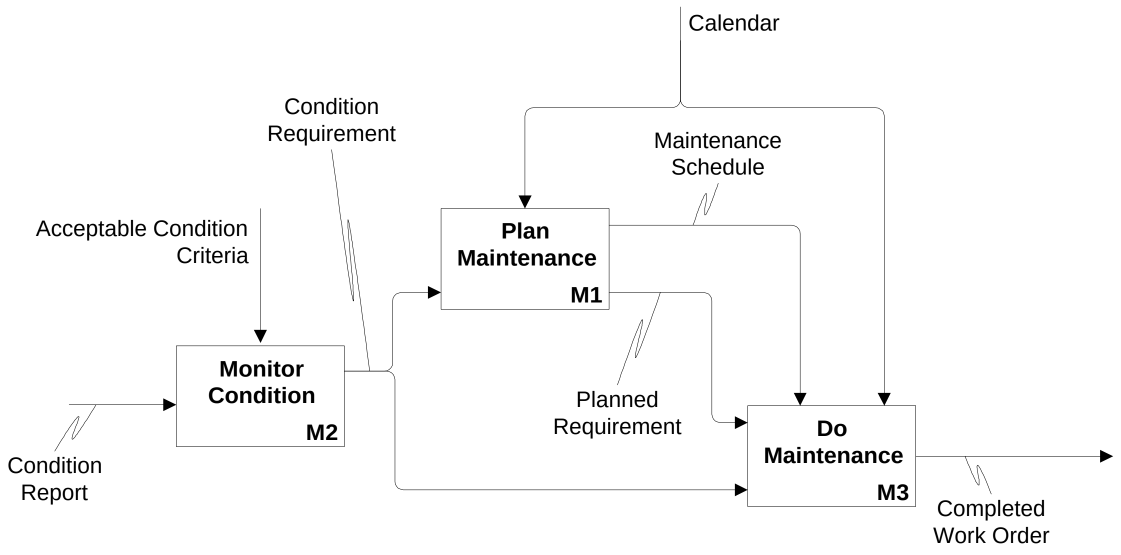
Multiple Activities

The output from one process may become the input to another process. Equally, the output of a process may indicate the need to carry out further work on a previous process. This is a feedback input to a process.

Processes may be in series, indicating that one is dependent upon another, or in parallel, indicating that there is no dependence.

It is conventional to draw process models from left to right and from top to bottom. Thus, the first process on a page is towards the top left and the last process to the bottom right. Staggering activities in this way enables outputs and feedback's to be easily identified and labeled.

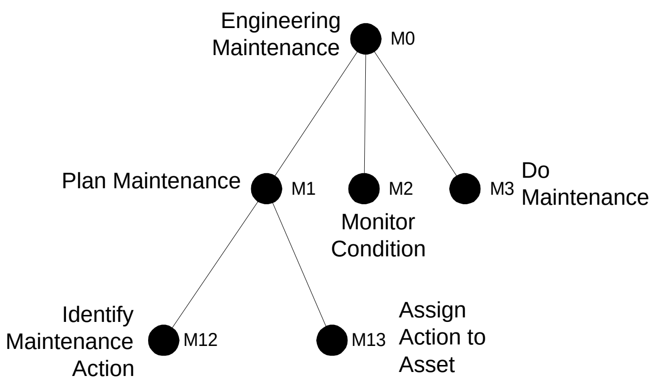
Arrows may be separated along their length. This allows a single input, output, control or mechanism to divide



to become an input to more than one process. Similarly, output arrows from more than one process can join together to become an input to a further process or to form the output from a model.

Decomposition

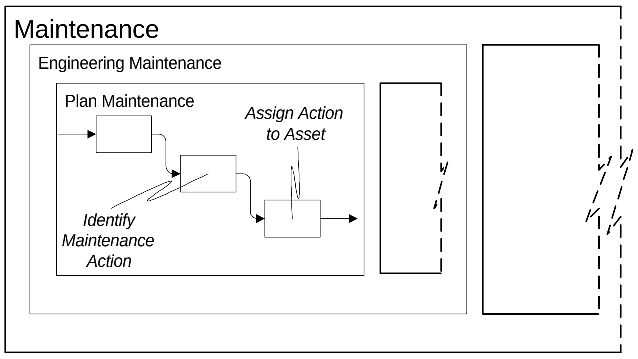
A complete process model may become very large. Large models can become difficult to understand and cannot be reproduced on normal size paper.



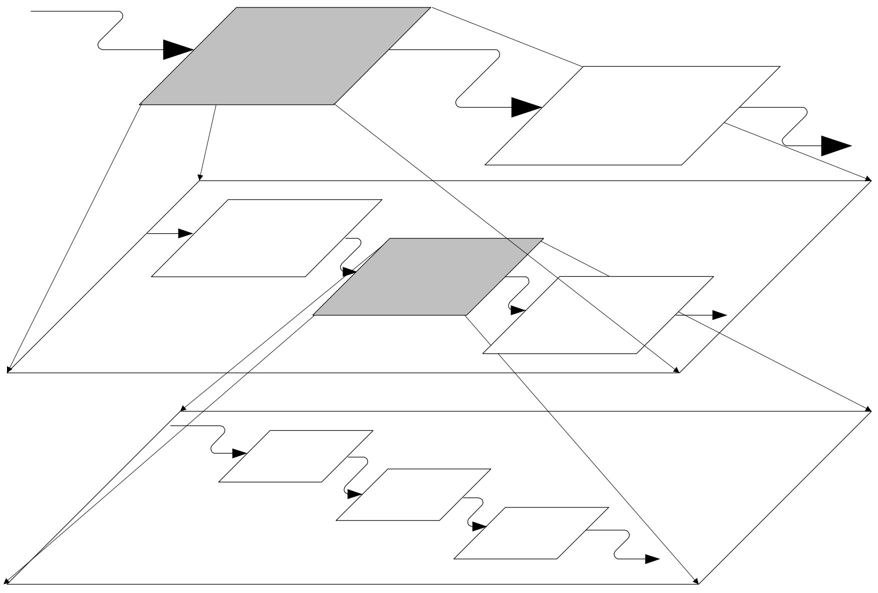
Decomposition allows a process to be described initially in a broad manner, e.g. Engineering Maintenance and then decomposed into sub-processes e.g. Plan Maintenance, Monitor Condition, Do Maintenance. As required, sub-processes may be further decomposed until further decomposition is not possible or necessary.

A process, when decomposed, has its own sub-processes. These do not take part in the decomposition of any other process.

Thus, there is no overlap between the decomposition of one process and the decomposition of another process.

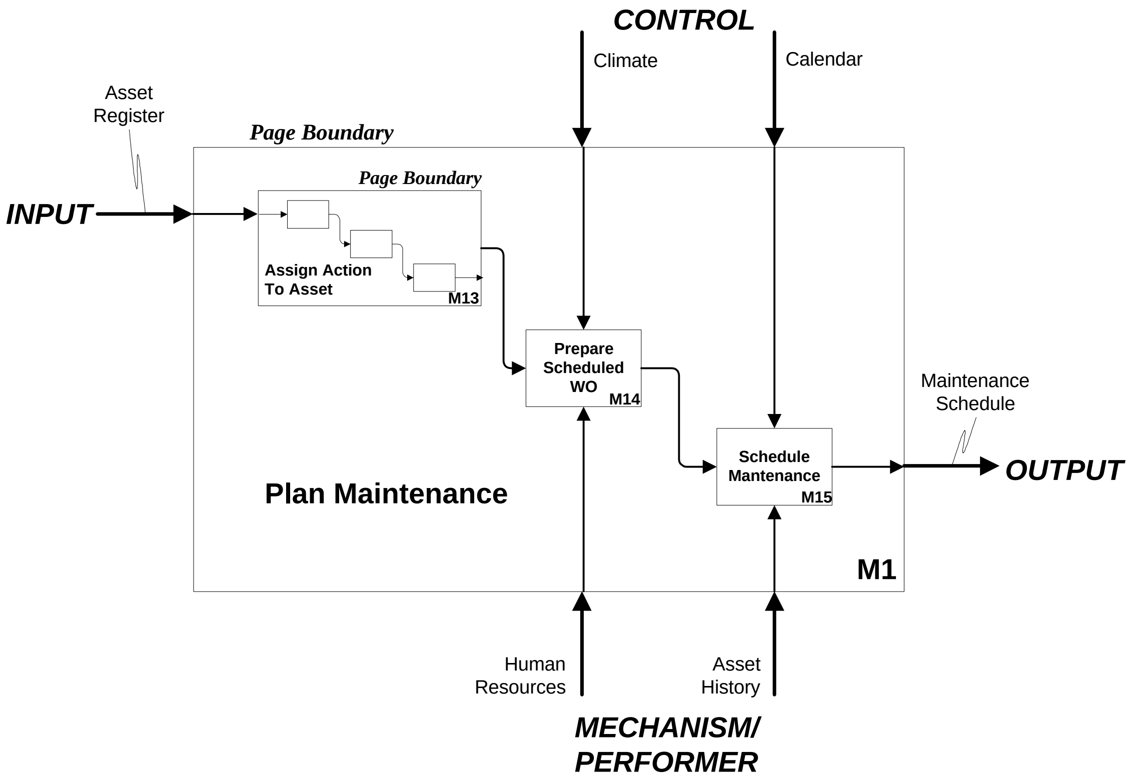


Each level of decomposition is shown as a separate process model on its own sheet. Typically, there is a minimum of three processes described on a sheet and a maximum of six. Three is considered to be a minimum that is relevant for decomposition of a process whilst six is considered to be the maximum that can be placed on a single sheet and remain clearly understandable. However, whilst these numbers represent a consensus on good practice, there are situations where a process may reasonably decompose into two sub-processes. It is more important that the decomposition of a process model should demonstrate the content of a business process clearly than that arbitrary rules are applied. Having said this however, it should be emphasized that placing more than six processes on a page will almost certainly result in reduced definition and clarity.



A process at the higher decomposition level is a ‘parent’ process and may be regarded as describing the boundary of the page on which the sub-processes at the lower decomposition level are drawn. The sub-processes are drawn on a ‘child’ model and are wholly contained within the boundary of the parent process.

Every input, output, control or mechanism that enters the parent process must have an equivalent on the child model (the exception to this is the use of tunneling which is described later). The reasoning for this can be



seen easily from the page boundary diagram below in which it can be seen that the actual process model really comprises all of the processes that are fully exposed at the lowest level of decomposition. Every parent process is, in fact, a convenient and easily understood process container. Therefore, the arrow that enters a parent process must have an ultimate destination that is a process at the lowest level of decomposition.

Decomposition Rules

There are few rules concerning the development of process models. Those that exist become obvious once stated. Some have already been indicated but are included below for completeness.

- ☞ **Each process label is unique.**
- ☞ **Each process has an identifier that is unique.**
- ☞ **A process is shown once and once only in a process model**
- ☞ **A process may decompose into sub-processes.**
- ☞ **Sub-processes should be shown on a separate ‘child’process model on its own sheet.**
- ☞ **The inputs, outputs, controls and mechanisms on a parent process must also be exhibited on the child process model.**

If a model appears to be 'too busy' because of the number of processes it contains, decompose the model further.

Title Block

Each sheet within a process model should be complete with a title block. This provides information about the model and locates the model with reference to other sheets.

At the minimum, the title block must show the following (normally at the bottom of the sheet):

- the name that is given to the model shown
  For a child model, this will typically be the name of the parent process;
- the node number of the diagram;
  This will be the identifier of the parent process;
- number of the model.
  This is a sequence number and is optional.

Additional information may be given in the title block. The following shows additional information that is used frequently with IDEF0 models and that is placed at the top of the sheet. This enables the following information to be displayed:

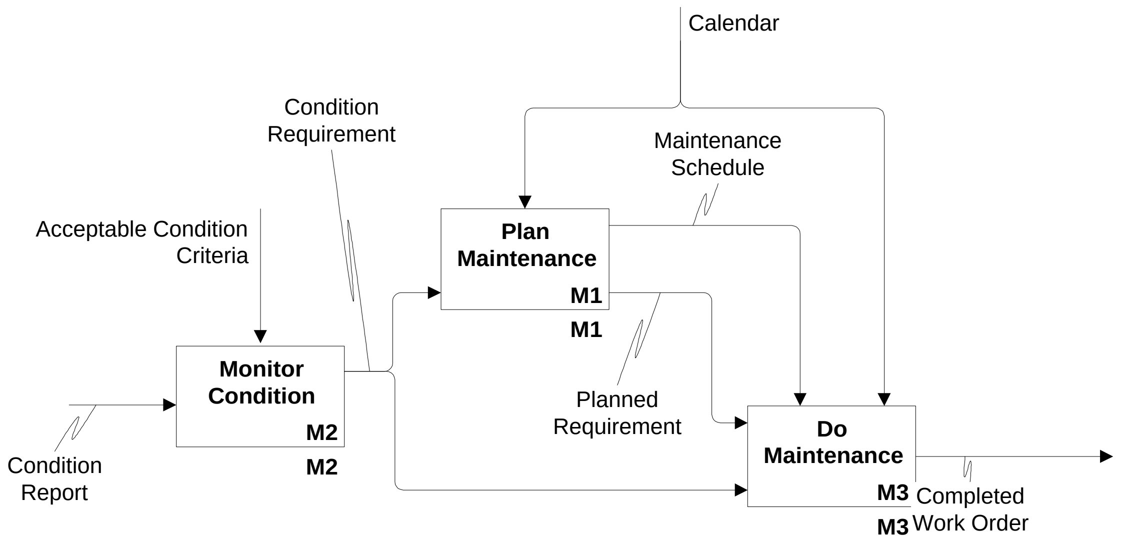
- the identity of the author of the model;
- the identity of the project to which the process model relates;
- the date on which the model was prepared;
- revision number of the model;
- where the model is to be used;
- identification of notes;
- status of the model (working/ draft/ recommended/ publication);
- the identity of the person authorizing the status;
- the date on which a status was authorized;
- the context of the child model (shown by marking the position of the parent process on a map of the process model at next higher level of decomposition).

USED AT:	AUTHOR:	J.D.Wix	DATE:	18.12.98	☒	WORKING	READER	DATE	CONTEXT
	PROJECT:	IFC Specification Development Guide	REV.:	2		DRAFT			□ ■ □
						RECOMMENDED			
	NOTES: 1 2 3 4 5 6 7 8 9 10					PUBLICATION			

Identifying Child Models

When a process is acting as the parent of a child model, placing the node number of the child model below the process box denotes this fact. If the node number of the child model is the same as the identifier of the parent process, then the situation will occur that the identity of the child model will be the same character sequence as the identifier of the parent process.

Many software tools enable hyperlinking between the identity of the child model shown adjacent to the parent and the child model itself, or for publication of the model as an HTML file in which elements of the model can be identified as hyperlinks to other files. These publication facilities are useful for navigation in the model.



Identifier Development

Process identification is normally developed by progressively adding a numeric character to the end of the identifier of the parent process.

Top Level

An alphanumeric reference with an initial alphabetic character followed by a single number e.g. A1,A2 etc. By following the rule of no more than six processes on a sheet within the model, the need to go beyond the number ‘9’ is avoided.

Child Model

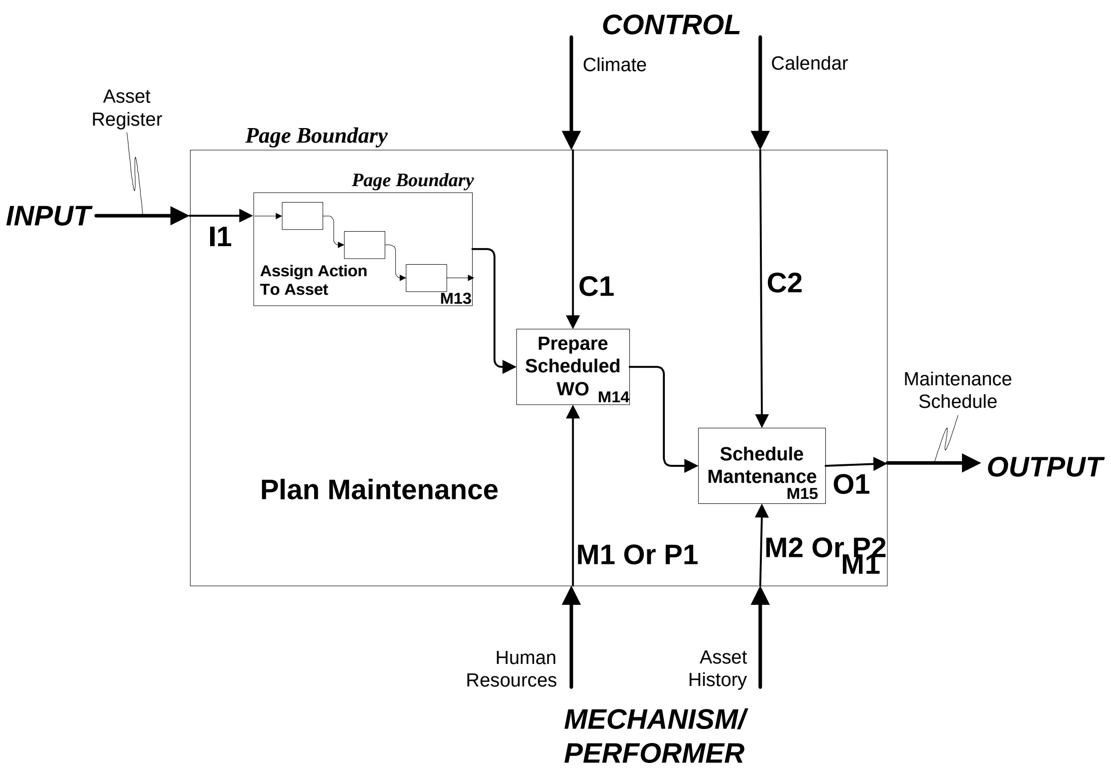
Add a further number to the identifier of the parent process. If the top level process is identified as A1 then sub-processes will be identified as A11, A12, A13 etc. whilst sub-processes of A3 will be A31, A32, A33 etc.

Subsequent Child Models

For each subsequent level of child model, add a further number to the identifier. Thus sub-processes of A31 will be as A311, A312, A313 etc. and sub-processes of A311 will be A3111, A3112, A3113 etc.

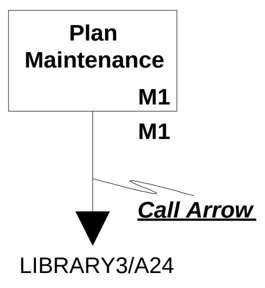
ICOM Notation

The need to label every input (I), output (O), control (C) and mechanism (M) on every child diagram can be avoided by using the ICOM notation. This allows the label on a parent process to be identified on the child model by an ICOM identifier.



To use ICOM identifiers, each inputs, output, control and mechanism must be in the same relative position on the child model as it is on the parent process. Then, starting from the left (for I and O) or the top (for C and M), the element can be identified by its character followed by a sequence number starting from 1. Thus, an input on a child model that is labelled I2 will be identifiable as the second input from the top on the parent process. A control that is labelled C3 will be identifiable as the third control from the left on the parent process.

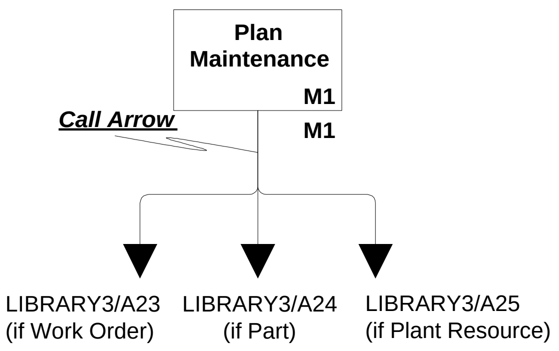
Call Arrow



A Call Arrow is a special type of mechanism that allows processes in a different part of the process model or even in a different process model entirely to be ‘called’. The arrow points downwards out of the bottom of the process box instead of upwards as is normal for mechanisms.

If a Call Arrow calls a process within the same process model, it is labelled using the node number of the model sheet on which the called process is located followed by the identifier of the process called. An oblique character (/) is used to separate the parts of the called identifier.

If a Call Arrow calls a process within a different process model, the identifier is the same as the above but is prefixed by the name of the model in which the called process is located.

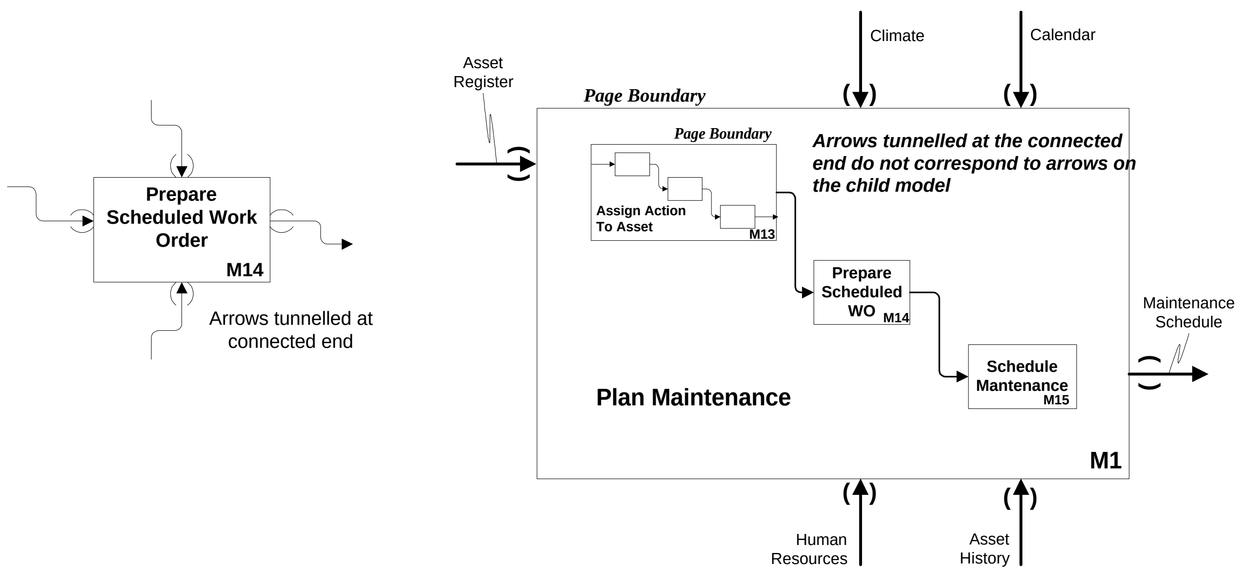

Plan Maintenance
M1
M1
Call Arrow
LIBRARY3/A23 (if Work Order)
LIBRARY3/A24 (if Part)
LIBRARY3/A25 (if Plant Resource)

When a process is called, only the parent process is identified. All child processes are the automatically called along with the parent.

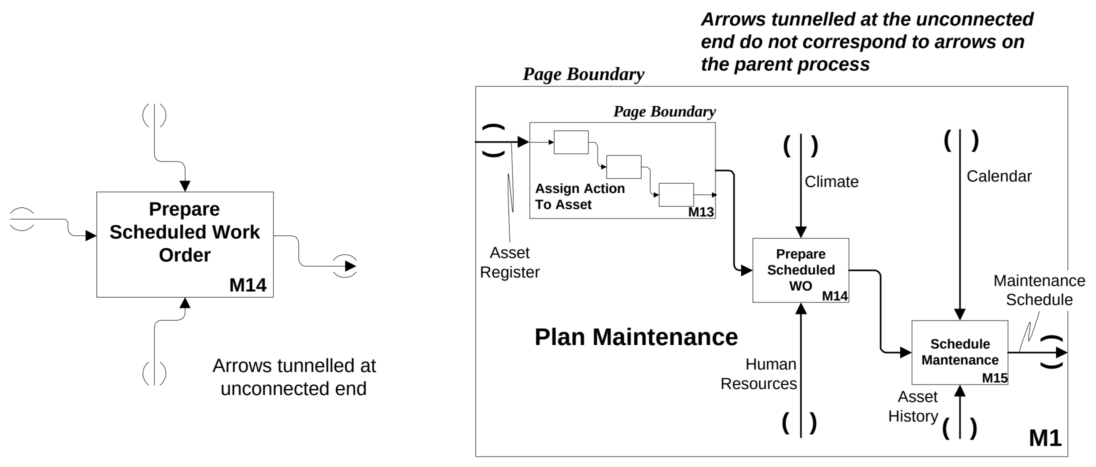
A Call Arrow can call several processes. However, only one can be used at a time. In order to select which of the available processes is to be called, a switching facility needs to be included on the arrow to the process. This identifies the conditions under which that particular process is called.

Tunneling

ICOM elements that are used for all processes in a child model can be omitted by identifying the fact that this is the case at the parent process. This is done by using a tunnel marking at the end of the arrow that connects into the parent process box. By using a 'tunnel' in this way, the author is stating that the arrow is a page boundary arrow that does not correspond to arrows on the child model.



Similarly, it is possible to omit ICOM elements at the parent process that are exposed in the child model. This is appropriate if the extent of detail at the parent process becomes too great. This procedure is again achieved by using tunnelled arrows but for this purpose the tunnel is placed at the unconnected end of the



arrow.

☞ **It is relatively simple to use arrows tunneled at the connected end to omit detail on a child model. However, caution should be exercised when using arrows tunneled at the unconnected end to omit detail at the parent process.**

Appendix D - Readers' Guide to EXPRESS-G

EXPRESS-G is a graphical modeling notation developed within STEP and used for IFC definition. It is used to identify classes, the data attributes of classes and the relationships which exist between classes.

EXPRESS-G is directly related to the EXPRESS data definition language. That is, everything that is drawn in EXPRESS-G can be defined in EXPRESS. However, not everything that can be defined in EXPRESS can be drawn in EXPRESS-G.

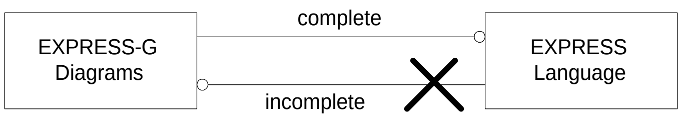


Figure D1 EXPRESS/EXPRESS-G relationship

This section provides a basic description of the EXPRESS-G notation as used in specifying IFCs, outlining the meaning and use of the various symbols. It is provided for information and to assist readers in understanding the graphical models used for specifying IFCs. It is not a complete reference to the capabilities of EXPRESS-G. Refer to ISO 10303 Part 11: EXPRESS language reference manual for the authoritative description of EXPRESS-G.

Classes

Things in which we are interested are known as classes. EXPRESS-G identifies classes in a rectangular box with solid lines enclosing the name of the class.

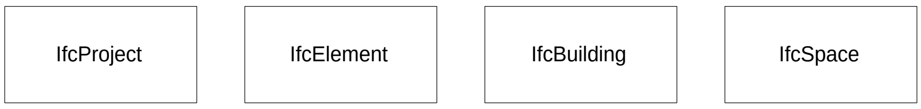
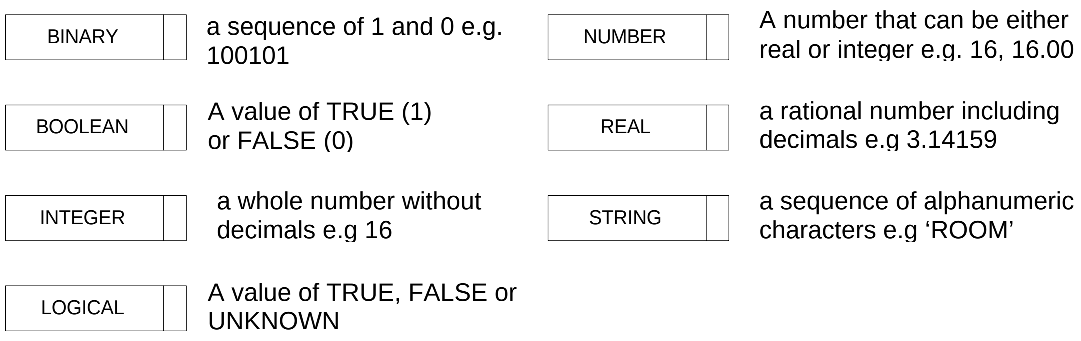


Figure D2 Entities

A class requires further information to describe it fully. Once a data specification has been developed for describing one instance of a class (an object), the specification can be generalized to cover all instances of the same class (objects).

Simple Data Types

Simple data types are the atomic parts of EXPRESS and EXPRESS-G; that is, they cannot be subdivided into anything smaller. A simple data type is shown as a solid rectangular box with a double vertical line at the right hand side of the box. The actual name of the data type is enclosed within the box.



BINARY	a sequence of 1 and 0 e.g. 100101	NUMBER	A number that can be either real or integer e.g. 16, 16.00
BOOLEAN	A value of TRUE (1) or FALSE (0)	REAL	a rational number including decimals e.g 3.14159
INTEGER	a whole number without decimals e.g 16	STRING	a sequence of alphanumeric characters e.g 'ROOM'
LOGICAL	A value of TRUE, FALSE or UNKNOWN		

Figure D3 Simple Data Types

Attributes and Relationships

Everything that is related to a class is considered to be an attribute or data member. It does not matter whether the attribute is a simple data type as above, a constructed or defined data type (see below) or another class.

Consider a class called IfcLayeredElement (this may be used to describe walls, floors and roof slabs as will be shown later). This may have attributes of TotalLength, TotalAreaPerSide and TotalVolume. These attributes may be either mandatory or optional. Mandatory means that whenever an instance of the class is used, a value of that attribute must be given. Optional means that a value may be given but that it is not necessary. In the example below, TotalLength is shown as mandatory whilst the other attributes are shown as optional.

Mandatory relations are shown by a solid line between class and attribute. Optional relations are shown by a dashed line between class and attribute. The name of the relation is written above the line. The circle shows the primary direction of the relation i.e. from IfcLayeredElement to IfcMaterialLayerSet.

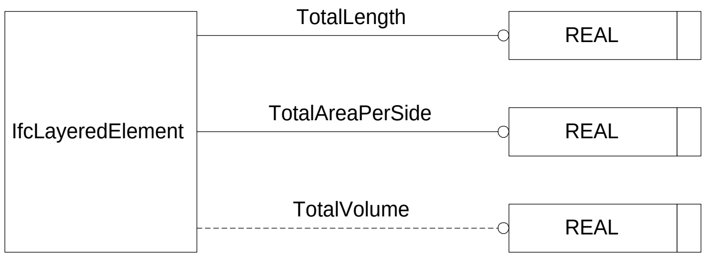


Figure D4 Mandatory and Optional Relations

Relationships Between Classes

Relations can also exist between classes. For instance, we could describe a relation between an IfcLayeredElement and a MaterialLayerSet (which describes the layers which make up the layered element) such that an ‘IfcLayeredElement has one MaterialLayerSet’.

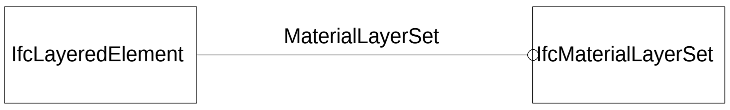


Figure D5 Relations Between Classes

EXPRESS does not allow the use of spaces in a class name or relation. In the above, the relation name has been made into a single word. An alternative is to use the underscore character to simulate a space.

However, this is not within the IFC naming convention for constructs being defined for IFC as part of the specification task. Constructs adopted from other external sources, most notably from STEP integrated resources, may have other naming conventions, currently those constructs uses underscore characters for entity, type and attribute names.

Sometimes, it may seem appropriate to use the same name for several relations that a class may have. This is not allowed in EXPRESS and therefore should never be used in an EXPRESS-G model.

Cardinality and Aggregation

Earlier, we saw how relationships could be described as mandatory or optional. These terms describe a numeric quantity to the relationship that is termed 'cardinality'. In the example shown, the mandatory relation identified that an IfcLayeredElement must have exactly one TotalLength whilst the optional relation identified that an IfcLayeredElement may have zero or one TotalVolume.

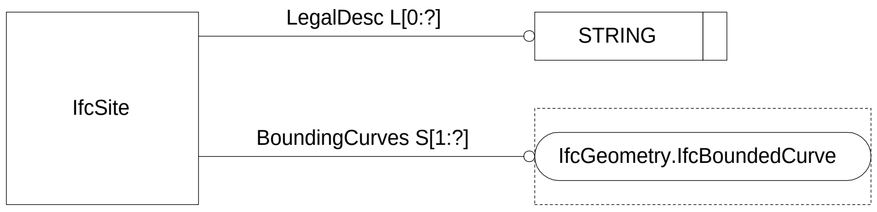


Figure D6 Cardinality and Aggregation

EXPRESS-G allows for relations of greater than one by providing various aggregation methods (or aggregate data types). For instance, a site may have zero, one or more legal descriptions and at least one boundary curve.

Aggregations allowed are:-

- ARRAY a fixed size collection of things with order represented as A[1:?].
- BAG a collection of things with no order and allowed duplication represented as B[1:?].
- LIST a collection of things with order represented as L[1:?].
- SET a collection of things with no order and no duplication represented as S[1:?].

The most frequently used aggregations will be SET and LIST. The site example shown uses list which indicates that legal descriptions must occur in a particular order with each list element being unique. If the relation was a SET, uniqueness would still be required but the elements could occur in any order.

The first character in square parentheses in an aggregation is the minimum possible value. The second character is the maximum possible value and may be either a number or the ? character which means indeterminate.

Supertype/ Subtype Relationships

Consider now the case where there is a general specification for a class but that this may be expanded by particular characteristics of subtypes of that class. For the layered element, wall, floor and roof slab have already been indicated as subtypes. Each subtype has all the general characteristics of the layered element which it acquires by INHERITANCE. However, each subtype may have additional attributes of its own.

Supertype/subtype relations are a special form due to this inheritance capability. Instead of writing "is_a" above the line (as in IfcWall is an IfcLayeredElement), a double thickness line is used.

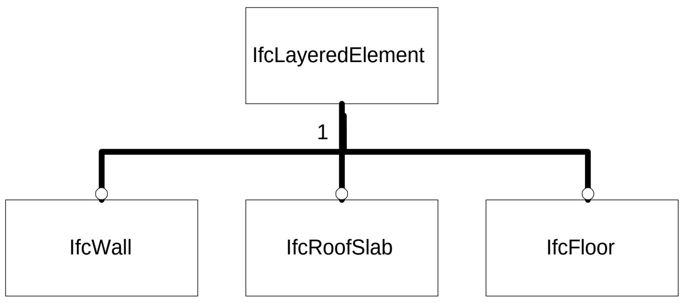


Figure D7 Subtyping

If we wish to make the subtypes exclusive, that is an IfcLayeredElement may be an IfcWall or an IfcFloor or an IfcRoofSlab, the number 1 is written at the branch of the relation. Subtypes may also be inclusive, that is an IfcLayeredElement may be both an IfcWall or an IfcRoofSlab at the same time, in which case the number is omitted. The choice of exclusive or inclusive subtypes is one of common sense.

In the example, the term (ABS) is used with the IfcLayeredElement to indicate that it is an abstract supertype. This means that it cannot exist in itself, only by virtue of its subtypes. Walls, floors and roof slabs are discussed on building sites but layered elements never are. However, the abstract supertype is useful since it enables attributes to be collected at a higher level within the data model and then inherited. Thus, it would be relatively easy to add a new subtype which would also inherit all the general attributes of a layered element.

Inverse Relationships

A complete relation between classes may need to be described in both the normal direction as described above and in the inverse direction. This can be done in EXPRESS-G.

In the example, an IfcZone has one or more IfcSpaceElements (which may be either a space or a zone). However, since there is no discrimination of zone type indicated such that a zone may be for security, fire alarm, HVAC etc., an IfcSpaceElement may be part of more than one IfcZone. In order to fully describe the relation between IfcSpaceElement and Zone, it must be shown in both directions with the inverse relation being identified by (INV) and placed below the relation line.

The example also shows the relation between building and zone since it is conceivable that a single zone might span several buildings.

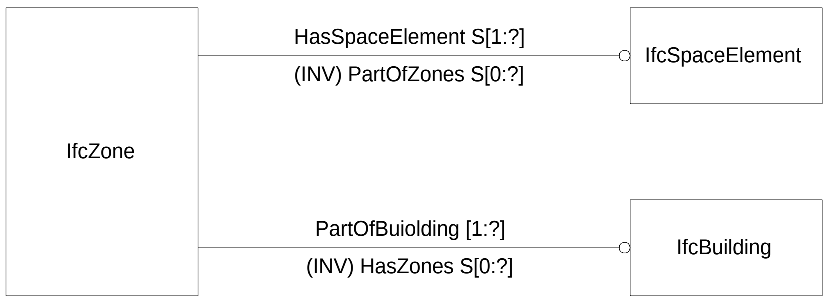


Figure D8 Inverse Relations

Enumeration Data Type

An enumerated data type provides for a range of possible values which the attribute may have described in an enumeration list. The attribute may only take one value from the possible range. It is shown as a rectangular box with dashed lines and a double vertical bar to the right. The name given to the enumeration is written in the box.

The example below shows an enumeration for IfcSpace which allows identification of the generic type of space. In this case, space types are enumerated as occupied, technical or circulation.

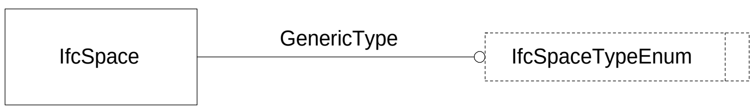


Figure D9 ENUMERATION Data Type

Select Data Type

A select data type enables the choice of which direction to follow in the model; in effect, select the class to be used for a purpose. In this sense, it is similar to a supertype/subtype and its construction in EXPRESS-G looks similar. A select is shown in EXPRESS-G as a rectangular box using dashed lines with a double vertical bar at the left hand end and the name given to the type written in the box.

In the example below, the IfcElementContainer enables consideration of either the site, the building, the building storey, or the space. Therefore any IfcElement can be (directly) part of one of these classes.

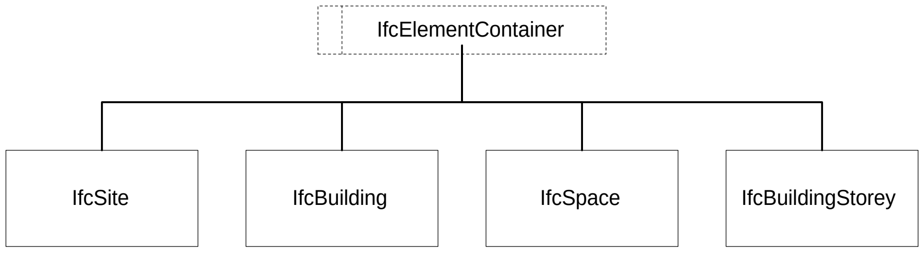


Figure D10 SELECT Data Type

Defined Type

A defined data type is used to take the place of a simple data type and is used to make the meaning of the model clearer. An organization may have a description which could take the form of a simple STRING data type. However, it might be more appropriate to make a data type called ‘text’ which could be used for the description. In this case, text is a type of STRING but is a more convenient way of handling a description. A defined data type is shown in EXPRESS-G as a rectangular box using dashed lines and the name given to the type written in the box.

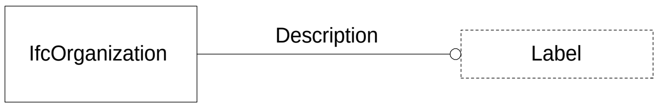


Figure D11 DEFINED Data Type

Page References

EXPRESS-G can be used for models spanning several sheets of paper. It does this by providing page reference symbols.

Where a relation crosses a page boundary, a means of identifying it is going to and where it has come from is required. These requirements are met by providing page reference symbols. Reference 'ONTO THIS PAGE' and reference ‘ONTO ANOTHER PAGE’ symbols are provided. Both of these symbols are rounded boxes with solid lines.

The 'ONTO ANOTHER PAGE' symbol incorporates the page number of the other page, an index number which must be unique for the page onto which it is connecting, and the name of the entity to which it is connecting. The target page and index numbers are separated by a comma.

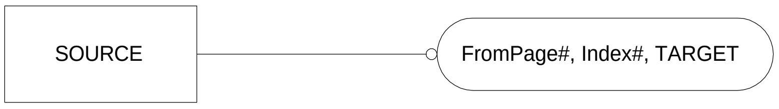


Figure D12 ONTO ANOTHER PAGE Reference

The 'ONTO THIS PAGE' connector includes in the box:

- the page number on which the reference entity occurs (which is the current page);
- the index number indicated on the matching 'ONTO ANOTHER PAGE' connector;
- the number of the page which originated the connection contained in parentheses (i.e. the page number on which the matching 'ONTO ANOTHER PAGE' connector exists).

Page Referencing can be used for supertype/subtype hierarchies as well as for aggregation relationships.

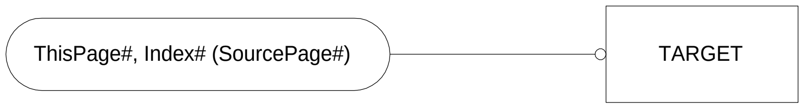


Figure D13 ONTO THIS PAGE Reference

Interfaces

An interface implies that there is a requirement in the current model to reference something from another model.

A class or data type that is referenced from another model is shown as a rectangular block using dashed lines containing a rounded box using solid lines. In the rounded box, the name of the model in which the class exists is declared. Following the model name, the name of the referenced class is given separated by a period (.). The referenced class may be given an alias in the current model (although this is not necessary) and this is written in the bottom sector of the rectangular box. The alias is the name by which the class is known in the current model.

In the example below, the fundamental IfcRoot class obtains a globally unique identifier and an owner history by interfacing with the utility resource schema.

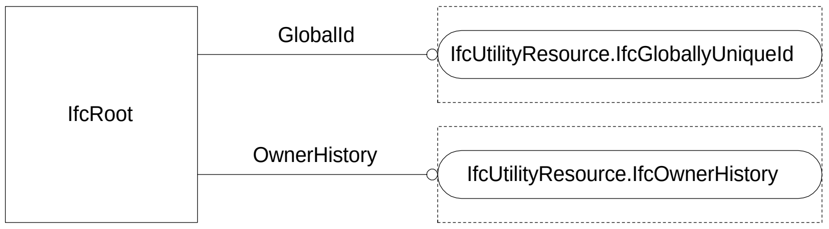

IfcRoot
GlobalId
IfcUtilityResource.IfcGloballyUniqueId
OwnerHistory
IfcUtilityResource.IfcOwnerHistory

Figure D14 Use of the REFERENCE FROM Interface

Domain Rules in EXPRESS-G

Domain rules allow constraints to be placed on attribute values or enable attribute values to be derived. The presence of an asterisk * against a relation name indicates that a WHERE rule applies.

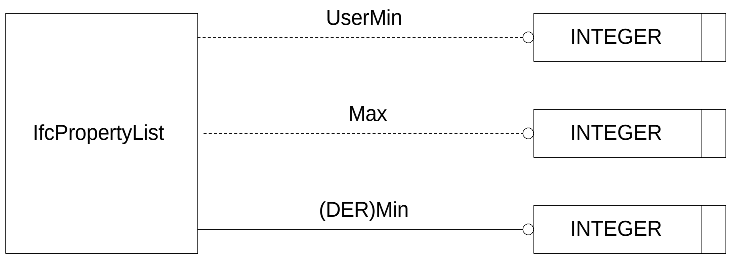


Figure D15 Indicating a Rule in EXPRESS-G

A (DER) placed prior to the relation name indicates the presence of a DERIVE rule. In the example above, the rule applied is that minimum number of properties in the property list is derived from the length of the list.

List of Symbols

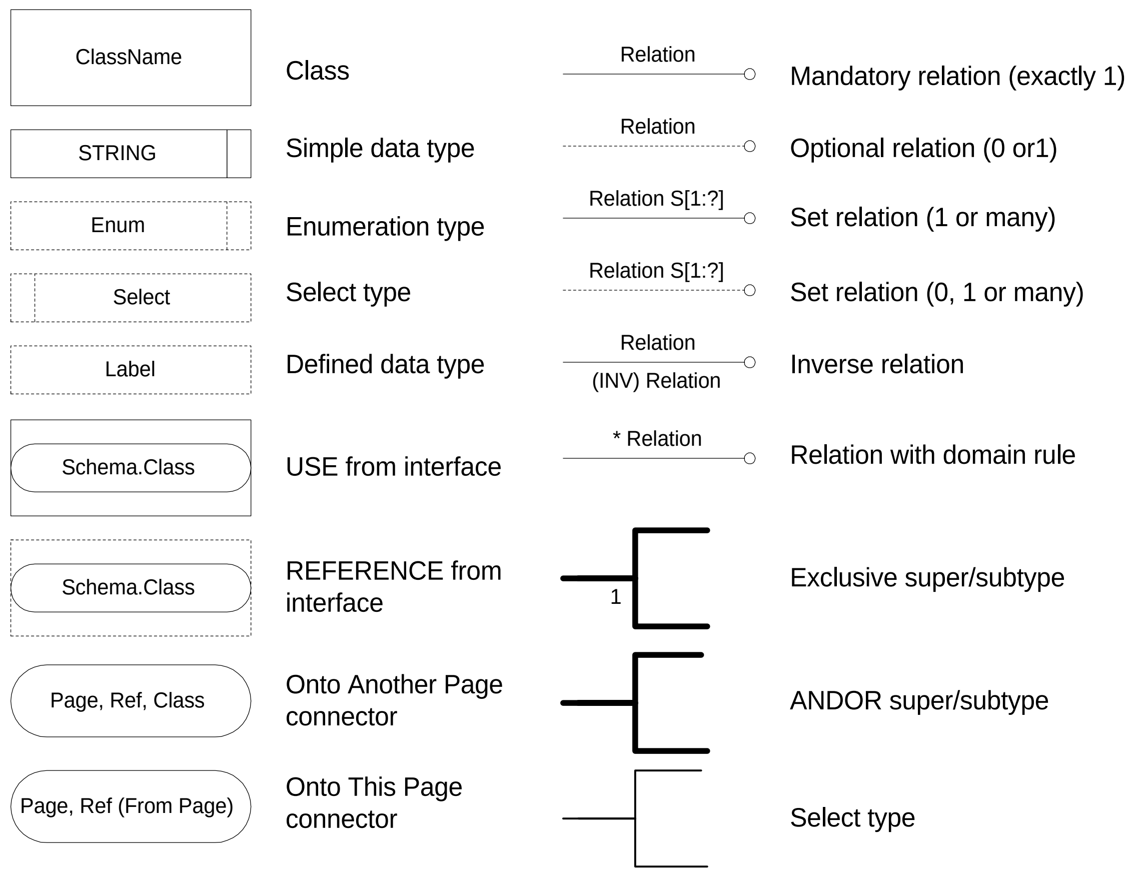


Figure D16 EXPRESS-G Symbol Legend

Appendix E - Readers Guide to EXPRESS

EXPRESS is a conceptual schema language which provides for the specification of classes belonging to a defined domain, the information or attributes pertaining to those classes (colour, size, shape etc.) and the constraints on those classes (unique, exclusive etc.). It is also used to define the relations which exist between classes and the numerical constraints applying to such relations.

EXPRESS Layout

EXPRESS uses the semi-colon character ; to terminate an expression. Whitespace and carriage returns are ignored in interpreting or checking the validity of an EXPRESS schema but they can be used to improve the layout of the schema so that it is human readable.

The following examples are exactly equivalent. However, the use of whitespace and carriage returns in the first example makes it much easier to read than the second example.

```
        ENTITY IfcClassification;
                ClassificationPublisher   : IfcString;
                ClassificationTable       : IfcString;
                ClassificationNotation    : IfcString;
                ClassificationDescription : IfcString;
        END_ENTITY;
```

```
ENTITY IfcClassification; ClassificationPublisher : IfcString; ClassificationTable : IfcString; ClassificationNotation :
IfcString; ClassificationDescription : IfcString; END_ENTITY;
```

Classes

EXPRESS is used to define classes. Within the class definition, all the attributes and behaviours which characterize it are declared. A class is declared by the keyword ENTITY and terminated by the keyword END_ENTITY.

An entity declaration creates a class and gives it a name. The declaration is terminated by END_ENTITY;

```
        ENTITY IfcOwnerID;
                ……
        END_ENTITY;
```

Attributes are the characteristics (data or behaviour) which are required to support use and understanding of the class. Attributes may be represented by simple data types (such as real, string, integer), or by other classes. Each attribute has a relationship with the class.

An attribute represented by a simple data type can be shown in EXPRESS by its data type.

```
        ENTITY IfcOwnerID;
                Identifier : IfcString;
                OwningApp  : IfcString;
                OwningUser : IfcActor;
        END_ENTITY;
```

Simple Data Types

A simple data type represents the atomic unit of data. Allowed simple data types are:-

REAL	Decimal numbers (e.g. 2.56). The decimal point must be present even if a number declared as REAL evaluates to a whole number equivalent to an integer (e.g. 2.0).
INTEGER	A whole number not containing a fraction or decimal element (e.g. 2)
NUMBER	A number which may arbitrarily evaluate to either an integer or a real, the specific representation not being important.
LOGICAL	A value which evaluates to TRUE, FALSE or UNKNOWN.
BOOLEAN	A value which evaluates to TRUE or FALSE only.
BINARY	A sequence of bits, each of which may have the value 0 or 1.

STRING	A sequence of characters. Case of the character is significant.

Attributes

An attribute represented by a relationship to another class is shown by the name of the relationship and the name of the class with which the relationship exists (in the direction of the relationship). Thus for an IfcLayeredElement with a MaterialLayerSet which is declared as a class in its own right, EXPRESS takes the form:-

```
ENTITY IfcLayeredElement
        …
        MaterialLayerSet : IfcMaterialLayerSet;
        …
END_ENTITY;
```

Cardinality Constraints

EXPRESS allows numerical relations to be mandatory or optional. A mandatory attribute which must be asserted is expressed by there being no prefix term before the attribute name as in the example above.
An optional attribute that may be asserted is expressed by the word OPTIONAL appearing as a prefix term before the attribute name

```
ENTITY IfcLayeredElement
        …
        TotalAreaPerSide : OPTIONAL IfcAreaMeasure;
        TotalVolume      : OPTIONAL IfcVolumeMeasure;
        TotalLength      : OPTIONAL IfcLengthMeasure;
        …
END_ENTITY;
```

Aggregation

A number of aggregation relations are supported. These include:

• ARRAY	a fixed size collection of things with order represented as A[1:?].
• BAG	a collection of things with no order and allowed duplication represented as B[1:?].
• LIST	a collection of things with order (sequence) and no duplication represented as L[1:?].
• SET	a collection of things with no order and no duplication represented as S[1:?].

```
ENTITY IfcLayeredElement
        …
        ImplGeoTopStartHeights : LIST [1:?] OF IfcLengthMeasure;
        ImplGeoTopEndHeights   : LIST [1:?] OF IfcLengthMeasure;
        …
END_ENTITY;
```

Within the IFC specification, LIST and SET are the most widely used aggregations.
The above example indicates that there must be at least one start height and one end height for a layered element but that there may be any number (the maximum limit is unspecified as identified by the ? character).
If the maximum number of list elements was limited, this would be identified by including the upper bound in place of the unspecified character:-

```
ImplGeoTopStartHeights : LIST [1:7] OF IfcAttLength;
```

Inverse Rule

Attributes explicitly capture a relation between classes and attributes Inverse relations can also be captured between a class and a named attribute of another class or between two classes.

```
ENTITY IfcProjectObject
        …
        ResultOf : SET [0:?] OF IfcProcessObject;
        …
UNIQUE
        UR1: OwnerID;
END_ENTITY;
```

```
        ENTITY IfcProcessObject;
        …
        INVERSE
                ResultsIn : SET[0:?] OF IfcProjectObject FOR ResultOf;
        END_ENTITY;
```

Note the use of the INVERSE keyword to define the inverse relation

Unique Rule

EXPRESS allows for the uniqueness of attributes to be defined by a 'unique rule'. This specifies that the value of an attribute which is declared to be UNIQUE is associated with only one instance of that class (object). Where more than one attribute is described as unique, each must be included in the UNIQUE declaration.

```
        ENTITY IfcProjectObject
        …
        UNIQUE
                UR1: OwnerID;
        END_ENTITY;
```

Derive Rule

In some cases, it is appropriate to include an attribute which can be computed directly from other attributes. This can be achieved through use of derived attributes which are declared following the DERIVE keyword. The following example also introduces the use of function IfcTotalWidth to calculate the derived value:-

```
        ENTITY IfcLayeredElement
        …
           DERIVE
                ImplGeoTotalWidth : IfcLengthMeasure := IfcTotalWidth (SELF.MaterialLayerSet);
        END_ENTITY;
```

Domain Rule

This is used to provide constraints on the values which attributes may have and is defined following the WHERE keyword. In the example below, the class can only exist if all three lists have the same number of members. This is used to ensure that those lists actually correspond, i.e. that for each thickness also an offset and a material is given.

```
        ENTITY IfcMaterialLayerSet;
                SetName         : OPTIONAL IfcString;
                Offsets         : LIST [1:?] OF IfcLengthMeasure;
                Thicknesses     : LIST [1:?] OF IfcLengthMeasure;
                Materials       : LIST [1:?] OF IfcMaterial;
           WHERE
                WR1 : (HIINDEX(SELF.Offsets) = HIINDEX(SELF.Thicknesses)) AND
                     (HIINDEX(SELF.Thicknesses) = HIINDEX(SELF.Materials));
END_ENTITY;
```

Arithmetical statements that are available within EXPRESS may be used in conjunction with the domain rule to provide constraints on attribute values. For instance, if the perimeter length of a window was constrained to be less than or equal to 4 metres, the rule would take the form:-

```
        ENTITY Window
                WINDOW_LENGTH       : REAL;
                WINDOW_HEIGHT       : REAL;
        WHERE
                perimeter : (WINDOW_LENGTH * 2 + WINDOW_HEIGHT * 2) <= 4.0;
        END_ENTITY;
```

Subtypes

EXPRESS allows for the classification of a class into subtypes. This defines a parent - child relation in which each subclass (referred to as subtype) contains more specific detail than its parent superclass (referred to as supertype).

```
        ENTITY IfcLayeredElement
```

```
                ABSTRACT SUPERTYPE OF (ONEOF
                        (IfcFloor,
                         IfcRoofSlab,
                         IfcWall));
                …
        END_ENTITY;

        ENTITY IfcWall;
                SUBTYPE OF (IfcLayeredElement);
                …
        END_ENTITY;
```

Note that the supertype declares the Wall as being 'ONEOF'. This indicates that the layered element is exclusively either a Wall or a Floor or a RoofSlab; it cannot be two or more at the same time. Alternative constraints on subtypes exist, most notably the ANDOR constraint that would allow the layered element to be either a Wall or a Floor or a RoofSlab or any combination of the three subtypes at the same time.

The current use of EXPRESS within IFC excludes all non ONEOF supertype constraints. Therefore no ANDOR or AND clauses will be found within IFC. EXPRESS supports both, single inheritance and multiple inheritance. Having more than one class within the SUBTYPE clause specifies multiple inheritance. IFC development does not use multiple inheritance.

A supertype may be included for the purposes of classification only, such situations occurring where it may be appropriate to include a supertype within the EXPRESS model for clarity. A supertype included for this purpose is never instanced; only its subtypes are used. In this case, it is known as an abstract supertype. The IfcLayeredElement is an example of an abstract supertype.

Declared Data Types

The type of a data item being used may be declared using a TYPE clause. For certain types of data item, this is necessary whilst for simple data types, it is optional. For instance, consider a space type that may be selected from an enumerated list of occupied, technical or circulation. The enumeration is declared as a data type as below:-

```
        TYPE IfcSpaceTypeEnum = ENUMERATION OF
                (Occupied, Technical, Circulation);
        END_TYPE;
```

Note that the use of the TYPE clause causes the declaration of the attribute within the class to be written in the same manner as if the relationship was with another class.

A SELECT data type defines a named collection of other data types. These may be other entities, a list of string values, a list of real values etc. As with enumeration’s, only one item from a SELECT list is used by an instance of the class which uses the TYPE.

```
        TYPE IfcBuildingSelect = SELECT
                (IfcBuilding, IfcBuildingStorey);
        END_TYPE;
```

Access to Other Schema

Classes that are defined in schema other than that which is current may be accessed using an EXPRESS interface. This allows schema to be partitioned into manageable parts and enables reuse of schema which may have been previously defined or defined elsewhere. There are two interface possibilities.
The REFERENCE specification allows declarations made in other schema (usually classes) to be referenced but does not make them part of the current schema i.e. the declarations remain remote.
The following indicates the referencing of entities defined in the IFC geometry schema.

```
REFERENCE FROM IfcGeometryResource
        (IfcCartesianPoint,         IfcBounded_curve,
         IfcPolyline,               IfcTrimmed_curve,
         IfcCompositeCurve,         IfcPlacement,
         IfcLine,                   IfcConic,
         IfcCircle,                 IfcEllipse));
```

Appendix F - IFC Properties and Property Sets

When you look at the IFC Object Model, what you see is a set of well defined ways of breaking down information into logical groups (the classes) and the structure of information that define the state of an instance of that class (the objects). The information structures provide a formal specification of attributes that belong to classes, define how data exchange and sharing using ISO 10303 parts 21 and 22 will be achieved and enable the specification of software interfaces using the Object Management Group's Interface Definition Language,

However, at any given time, the model is not complete. There are many types of information that users might want to exchange that are not currently included within the IFC Object Model.

Additionally, the inclusion of classes of information beyond a certain level could cause the formally specified model to grow to such an extent that it could become difficult to manage and implement.

For many classes that exist within the model, it is possible to define 'Types' of an element. By defining these Types, it is possible to create standardized ways of describing information without the need for the formally specified part of the model to grow.

Frequently, there is a need to extend the attributes that are attached to an individual object or group of objects. Yet it may not be necessary to extend the attributes for every object within the same class. Using the same capabilities as for Types of an element, it is possible to define such sets of attributes and associate them with individual objects.

This is done using the Property Definition model. This allows classes and their attributes to be defined and attached to objects and relationships as objects when needed and not before.

Formally, the Property Definition model provides a meta-model of how to define such classes and attributes (a meta-model is a model that tells you how to develop a model). These classes are termed Dynamic classes (as opposed to the Static classes defined in the formal model) and extend occurrences of the static classes via predefined relationships at 'run-time' (that is, when you are actually working with them).

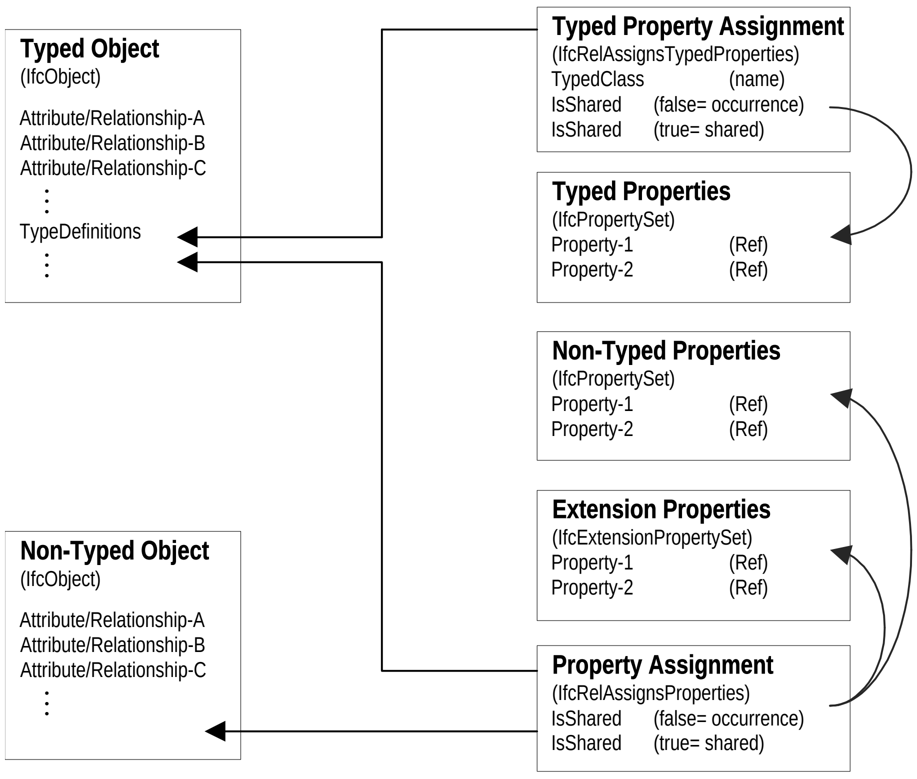
One motivation for defining a Type of an element is to establish a standard that will be used many times in a project. In these cases, a standard Type is established through the definition of a set of properties that are constant for all occurrences of that Type in the model. That is, there will be a single record for the dynamic class and its attributes in an exchange file.

Another motivation for defining a Type of an element is to establish a use or purpose for the element that requires a standard set of properties be defined for each occurrence. In these cases, this standard set of properties will be determined by the Type, but values for these Properties will vary for each occurrence of the element.

Thus property definitions can be either:

- Type defined and shared among multiple occurrences of an element, or
- Type defined but specific for a single occurrences of an element, or
- Non type defined but within IFC specifications and assigned to objects, or
- Extension definitions by end users that are not within IFC specifications.

The following figure gives an overview of the different usage of properties in the IFC Object Model.



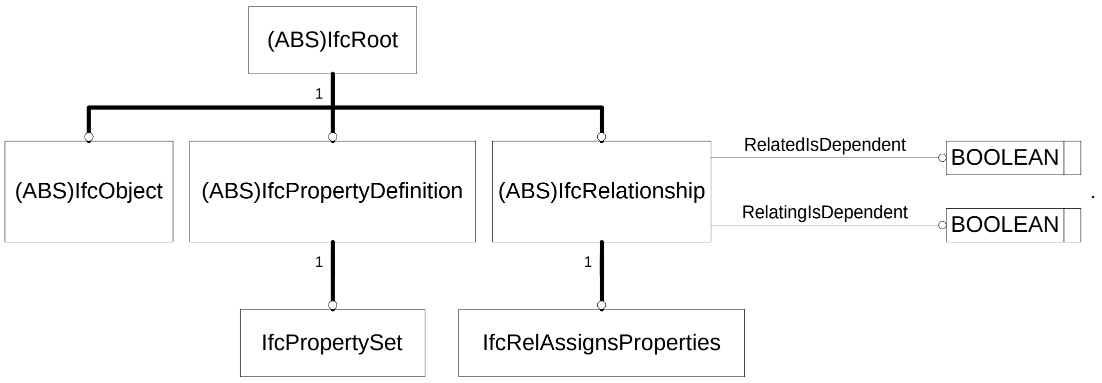
The remainder of this section looks at the Property Definition model and describes, on a class by class basis, how it operates.

Note that the first part of the Property Definition model is defined within the IfcKernel schema. This part deals with the structuring and assignment of properties. The second part of the Property Definition model is defined within the IfcPropertyResource schema and this deals with the various types of property that can be defined.

Properties at the Highest Level of the IFC Model

This section is contained within the IfcKernel schema)

The fundamental class in the IFC Object Model is the IfcRoot class. Every class in the model inherits basic attributes from this class (except for the classes in the Resource layer which are independent). Therefore, every object possesses the attributes of ifcRoot. These attributes include a unique identifier that remains with the object throughout its existence and an owner history that identifies when it was created and who by and who is its current owner.



Therefore, every property definition has an identifier and a history.

There are three classes defined as subtypes of IfcRoot and these are the fundamental structuring classes within the IFC Object Model.

1. IfcObject

   Provides the basis for all classes that are statically defined within the IFC Object Model. That is, all attributes for the class are defined explicitly within the model and are visible in the formal EXPRESS code.

2. IfcPropertyDefinition

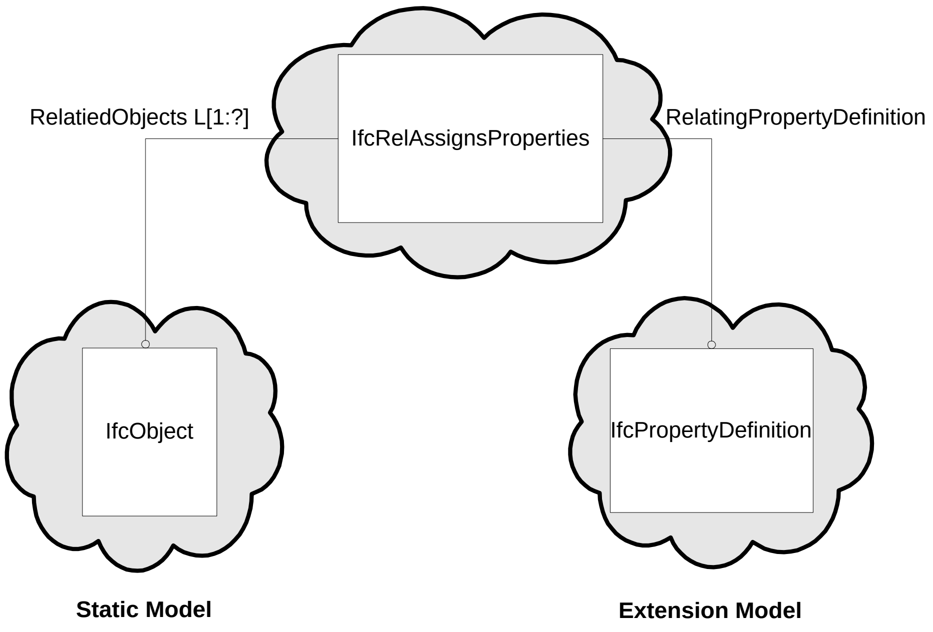
   Provide the basis for all classes that are dynamically defined within the IFC Object Model. That is, all attributes for the class are defined implicitly within the model and are not visible in the formal EXPRESS code.

3. IfcRelationship

   Provides a means of relating objects to objects or property definitions to objects by means of various predefined types of relationships such as grouping, nesting, assembly etc. This class has two attributes that are of type Boolean (that is, they can only have the value TRUE or FALSE). These attributes identify whether the class that has the ‘Relating’ attribute depends for its existence on the class that has the ‘Related’ attribute or vice versa. In practice, only one of these attributes can have the value TRUE; the other attribute has the value FALSE by definition.

   In the case of an IfcPropertyDefinition, the attribute ‘RelatingIsDependent’ must have the value TRUE because the existence of the property definition in association with an object is dependent on the existence of the object to which it is to be related.

Extending Objects



The purpose of the Property Definition model is to provide means of extending the information available about objects. This could be any type of object (class). The means of relating an IfcPropertyDefinition to an object is via the IfcRelAssignsProperties class which is one of the predefined types of relationships within the IFC Object Model. As the name implies, it has the single purpose of associating properties with objects.

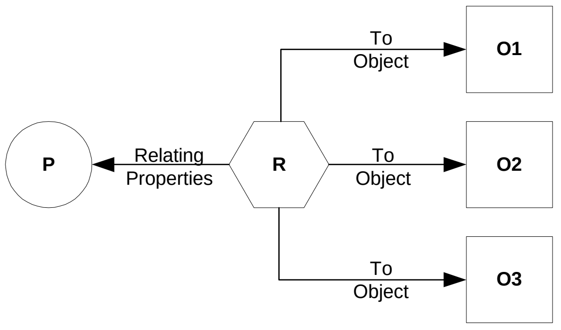
Although the relationship identifies that a property definition is associated with an object, the IfcPropertyDefinition class is an abstract supertype which means that it is never used in itself. It is objects of the IfcPropertySet subtype that are actually used.

Consider that there is a Property Definition P that is to be related to objects O_1, O_2 and O_3 using relationship object R such that:

P is related to O_1 by R

P is related to O_2 by R

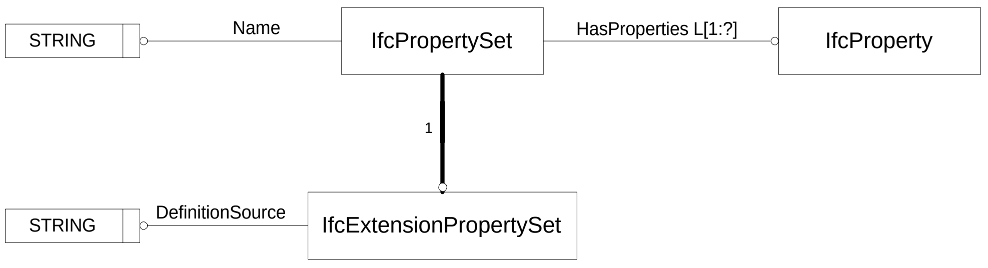
P is related to O_3 by R



Property Set

The IfcPropertySet class is the operative subtype of the IfcPropertyDefinition. It is the IfcPropertySet that appears in an exchange file and not the IfcPropertyDefinition.

An IfcPropertySet is a container that holds collections (or lists) of properties.



The fundamental aspect of the IfcPropertySet is that it contains a list of properties. It must contain at least one property and may contain as many as are necessary. Each property in the property set must be unique.

Since a property may be simple, the property set may define zero or many attributes such as AirFlowRate or ResistanceToFlow.

Because a property may be an object reference, the property set may define zero or many references pointing to an object defined in the static part of the IFC Object Model. For instance, if the installation cost of a centrifugal fan needs to be known as part of the property set, a reference to an IfcCost object defined in the IfcCostResource schema would be used.

Because a property may be a property list, the property set of current interest may contain references to zero or many lists of properties (which in turn may reference other property lists). This allows for nesting of properties, which provides an extremely powerful capability for dynamically extending the information content linked to an IFC object.

Type Relationships

This section is contained within the IfcKernel schema

The IfcRelAssignsTypedProperties class is a special type of the IfcRelAssignsProperties class. It enables a property set to be 'type defined' and then related to an IfcObject that has a special 'type' attribute.

Type definition enables dynamic or runtime definition of objects. Such type definitions enable:

- Relating of an object Type, for which a set of properties is defined that are attached at runtime. This is done though relating one or more IfcPropertySets.

  For instance, there may be a class called IfcFan within the static model but the different types of fan that may exist (single stage axial, multi-stage axial, centrifugal, propellor etc.) are not in the static model. These are declared as types of the IfcFan through a type relationship attached to the IfcFan class. Each type of fan that could be defined in IFC is included in an enumeration of fan types. The "GenericType" attribute on IfcFan is of this data type. Therefore, an IfcFan's Type is setting this GenericType attribute (selecting from the enumeration of Fan types).

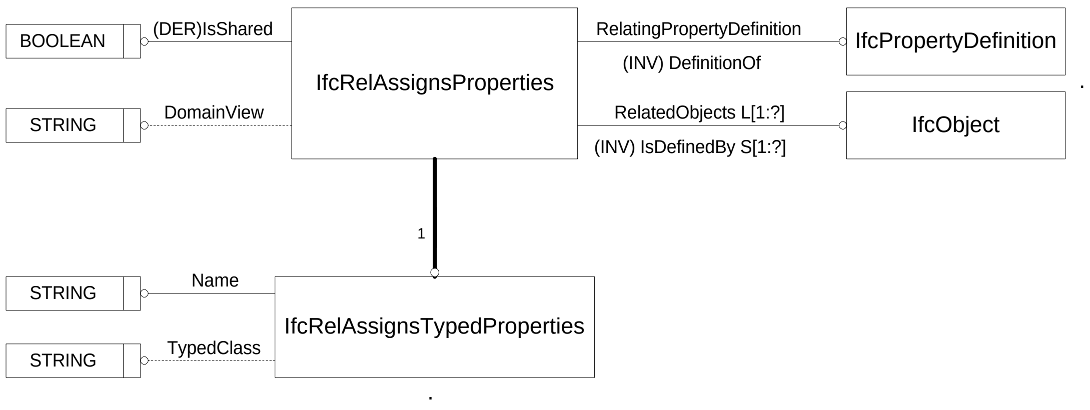
- Sharing a standard set of property values defined in a publicly accessible IfcPropertySet across multiple occurrences of that object type.

  For instance, a standard range of properties with known values might be defined for the maintenance of centrifugal fans. These properties will be applied to every centrifugal fan and do not have to be copied to very instance of that Type of object.

- Defining different property values within a private copy of the IfcPropertySet for each instance of that object type.

  For instance, all centrifugal fans deliver a volume of air against a known resistance to airflow. Although these properties are assigned to every centrifugal fan, the values given to them differ for every instance.

An object that can have a related type defined property set has a type attribute which is always an enumeration list with the name xxxTypeEnum. The values contained in the enumeration list indicate all of the typed classes that can be related to the object. This is reflected by the 'TypedClass' attribute of the IfcRelAssignsTypedProperties class which has a matching value to one of the items in the enumeration list.



The model allows for several type relationships for a single object. This is achieved by using the inverse SET [1:?] aggregation from the IfcObject to the IfcRelAssignsProperties class.

A major advantage of using a typed property definition is that the Type can be changed (since the Type is referenced). It can be changed at runtime whenever is the user considers it necessary or appropriate. This offers the possibility that the object can evolve throughout its lifecycle, changing the attached properties (growing or shrinking them) to reflect its current state.

In the property definition example shown above, the property definition (as a property set) is shared amongst several objects by the relationship object. Equally, it could be applied to a single object. The context of whether or not it is shared is identified in the IfcRelAssignsProperties class by the attribute *IsShared*. This is a BOOLEAN value (TRUE/FALSE) whose value is derived by context. If there is only one related object, the value is FALSE. If more than one related object exists, then the value is TRUE.

Property sets may be defined from different domain/application points of view. The *DomainView* attribute defines the domain view for which a property set is assigned to an object. For instance, some attributes of a centrifugal fan might need to be seen by a building services engineer whilst others might need to be seen by a maintenance contractor (as part of an FM requirement). These would be defined as separate property sets, each being assigned to the object with the value of the domain view attribute being specified as ‘BuildingServices’ or ‘FacilitiesManagement’.

The *TypedClass* attribute gives the name of the class within the static part of the IFC Object Model being typed. In the example identified above, the typed class attribute would be given the value IfcFan to indicate that it is this class within the IFC Object Model to which the property definition belongs.

The *Name* attribute gives the name of the type being defined. In the example identified above, the name attribute would be given the value ‘CentrifugalFan’. Note that this value MUST match the GenericType attribute of the IfcFan object being typed.

Extension Properties

This section is contained within the IfcKernel schema)

An IfcExtensionPropertySet allows for the definition of a set of properties that are not specifically tied to an object by a type definition and that are not published as part of the IFC specifications. Such properties may be defined and then attached to any appropriate object via the property assignment relationship object.

Where an extension property set is defined from a published source, the *DefinitionSource* attribute enables the identification of that source.

Extension property sets could be particularly valuable for projects that wish to extend the range of objects and attributes within the IFC Object Model for particular reasons. In this case, the definition source attribute may be used as a pointer to the project participant responsible for defining the extension property set.

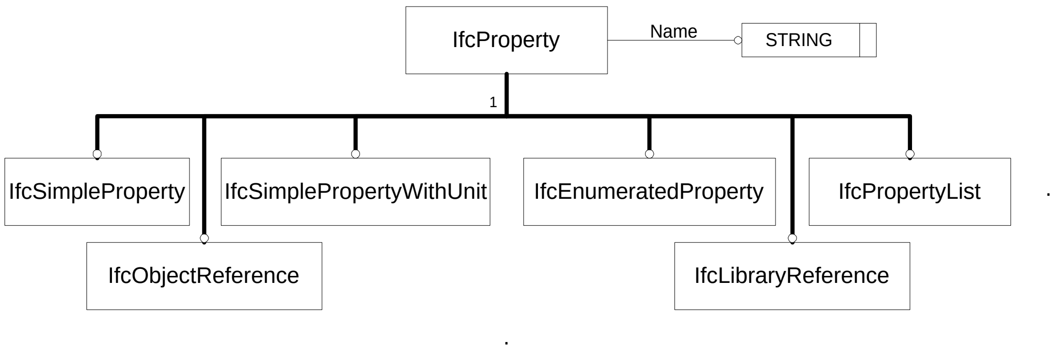
Use of extension property sets in a local, regional or project specific way is a powerful addition to the IFC Object Model capability. However, because the schema of the property set is not declared within the model or as part of the IFC published specifications, their use in this context does require that a convention be adopted that is known to all participants that are likely to receive such information.

In many ways, this might be considered analogous to the use of layers and the definition of layer conventions in CAD systems. However, it does require more precision than a layer convention which only identifies groupings of graphical entities; extension property sets can define the characteristics of individual objects.

Property

This section is contained within the IfcPropertyResource schema)

The IfcProperty is the common abstraction for all Properties defined within the IFC Model. Those Properties can be either simple properties (a single attribute with a single value) either with or without units, references to objects defined in the static part of the IFC Object Model, lists of properties or references to sets or lists of properties external to the IFC Model.

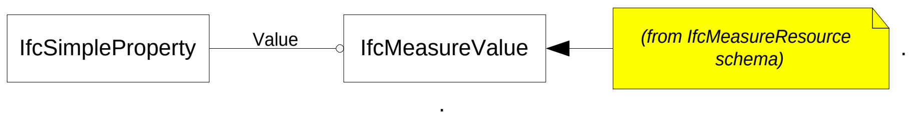


Every property must have a *Name* attribute that identifies it.

It is to be expected that a dictionary of standard IFC properties will be defined progressively as use of the dynamic part of the IFC Object Model expands. At present, properties are arbitrarily selected and thus there is a possibility that the same property may occur under different names[1].

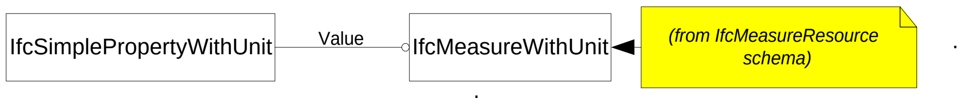
Simple Property

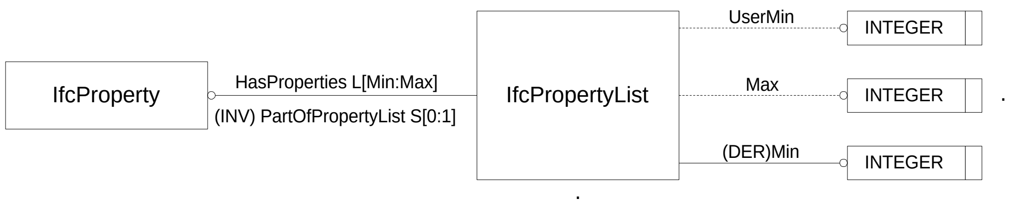
A simple property is a single attribute that has a *name* -- *value* pair. The value, in the case of IfcSimpleProperty, is defined individually for the attribute by an IfcMeasureValue which may exist without units. The IfcMeasureValue class is defined in the IfcMeasureResource schema.



Simple Property With Unit

A simple property with unit is a single attribute that has a *name* -- *value* pair. An IfcMeasureValue defines the units for the value, in the case of IfcSimplePropertyWithUnit, individually for the attribute. This defines the unit type for the attribute uniquely. The IfcMeasureWithUnit class is defined in the IfcMeasureResource schema.





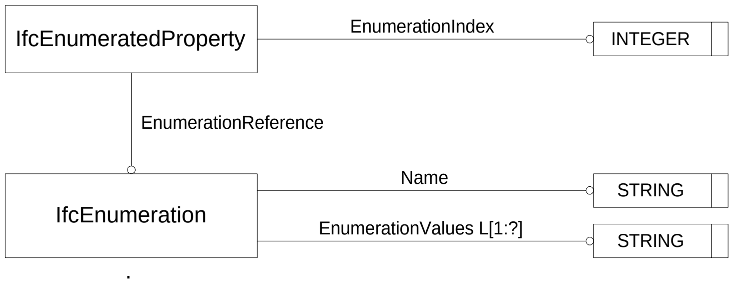
Property List

A property list provides the means for a property set to contain more than one property. The list can contain any of the types of property that are defined within the IFC Object Model including other property lists. This class therefore provides a means of nesting properties through more than one level.

[1] Although this is possible, a part of the integration role of the IAI Specification Task Force is to trap and resolve such problems. The dictionary will become important when the number of properties grows large.

Enumerated Property

The enumerated property class provides the means for a list of possible values of the property to be provided



from which only one can be selected at that particular point in time. At a future time, it should be possible to change the value selected by referencing a different value within the list. This is done through the use of the enumeration index that has an integer value whose maximum value is less than or equal to the number of values in the list.

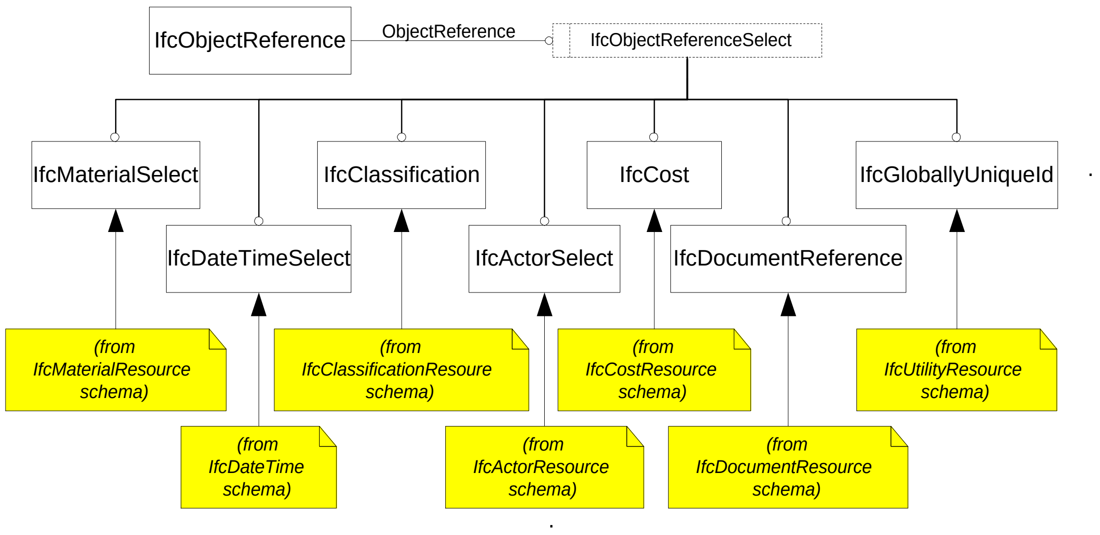
IfcEnumeration

The IfcEnumeration class holds the list of values into which the enumeration index points. Additionally, the enumeration has a name by which it can be identified and that identifies it so that it can be defined as part of the IFC specifications.

For example, an IfcEnumeration might have the name 'BladeCurvature' from which the configuration of the impeller blades in a centrifugal fan will be selected. It might have the values ('Forward', 'Backward', 'Radial', 'Other'.). The selection made might be backward curved blading in which case the enumeration index would point to the second item in the list. At some point in the future, blade curvature might be changed to radial (i.e. no curvature of the blade) because it offers a better performace for the required duty. In this case, the enumeration index would be changed to point to the third item on the list.

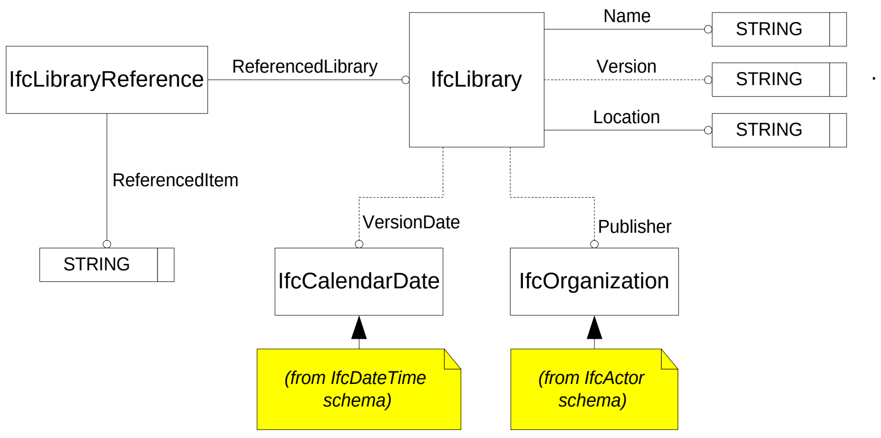
Object Reference

An object reference enables reference to objects whose structure is defined by the static part of the IFC Object Model. Specific types of object may be referenced according to their identity within an exchange file (shown by the object types from the static part of the IFC Object Model below). Alternatively, the object reference may be by a globally unique identifier. This means that the reference can only ever be to that one object.


IfcObjectReference
ObjectReference
IfcObjectReferenceSelect
IfcMaterialSelect
IfcClassification
IfcCost
IfcGloballyUniqueId
IfcDateTimeSelect
IfcActorSelect
IfcDocumentReference
(from IfcMaterialResource schema)
(from IfcClassificationResoure schema)
(from IfcCostResource schema)
(from IfcUtilityResource schema)
(from IfcDateTime schema)
(from IfcActorResource schema)
(from IfcDocumentResource schema)

Library Reference

The objective of the Library Reference is to enable an organization that provides information to make it available according to the structure of a property set[2] defined as part of the IFC Object Model. Providing that this is the case and providing that this information is accessible to both the sender and receiver of an IFC based information stream, then the property set can be referenced by a location (that might be a URL of another form of location address).



A library reference contains a pointer to the external library that contains the information to be referenced through the *ReferencedLibrary* attribute.

The *ReferencedItem* attribute identifies the particular item within the Library that contains the information to be referenced. In this sense, it may be considered as the key to obtaining the information at a time when it needs to be exported into the actual IFC compliant format.

IfcLibrary

The IfcLibrary class enables identification of the actual external library that is to be referenced via an occurrence of the IfcLibraryReference class. A given library may be referenced by many occurrences of a library reference.

Name is the name by which the library is normally known.

The *Version* attribute identifies the version of the library that is referenced. Although optional, this attribute is important. Information within a library may be subject to continuous updating so that, after the reference is made to a referenced item in one version of the library, it may be updated in a later version. The version attribute may be used to identify whether the reference is to the current library or a previous version.

Each version has a *VersionDate* that identifies the date on which the current version of the library was issued. This attribute may also be used to ensure that references are to the current version of the library. The attribute type is a calendar date that is defined within the IfcDateTime schema.

The *Location* attribute identifies the place at which the library can be found. This is, effectively, a fully qualified address. In the case of a library that is accessed via the World Wide Web, it is the URL of the library.

A library has an organization who acts as the *Publisher*. Definition of organization is given in the IfcActor schema.

[2] This is a first stage in developing the capability to fully access information libraries that are accessible to multiple users. The objective is to lessen the amount of information needing to be exchanged because it is as available to the receiver as to the sender.
At this stage of development however, it is recommended that the use of the Library Reference Property is limited to property sets that do not use nesting or object references i.e. they contain only simple properties or property lists.

Converting a Library Reference to a Property Set

It may become necessary to embed the information contained within the external source into the project model. This requires that the property set be read into the project model from the external source and converted into the appropriate IfcProperty subtype (e.g. simple property or property list).

Referencing Multiple Libraries

There will be times during the development of the AEC/FM process for a particular project when the performance information associated with an object will be known but that the particular technical solution to be adopted has not yet been decided on. There may be several technical solution possibilities, each of which could be satisfactory. In this case, it might be appropriate to store all of the library references from which a technical solution might be selected and to assign these to the object.

This can be done using a property list in which all of the properties contained in the list are library references.

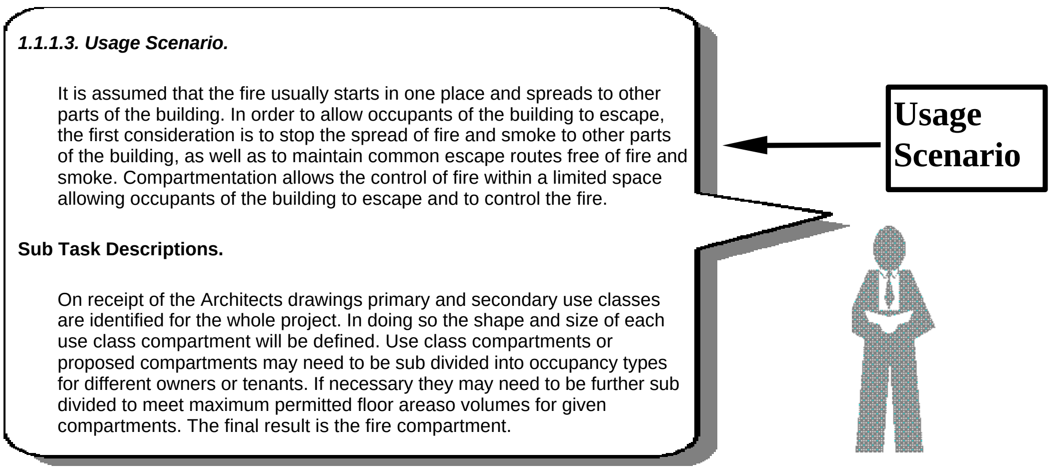
Alternatively, a property set could be specified for support of library references that contained a property list that, in turn, contained the library references. This has the added value that it could be shared amongst many instances of a class.

Appendix G - Example Walkthrough

The following presents a series of slides used during a review meeting to consider the architectural process of ‘means of escape’. The review concerned validation of the process model developed at the preliminary stage. It is presented here to provide guidance on developing a sequence to information presented during a review Walkthrough that is also sufficiently complete and self explanatory so as to be used for distance review.

1: Review and Validate Usage Scenario

Ensure that there is a consensus that the usage scenario is clear, comprehensible and states known ideas and facts about the area of interest in a manner which can be understood by a modeller or software implementer.


Usage Scenario

1.1.1.3. Usage Scenario.

It is assumed that the fire usually starts in one place and spreads to other parts of the building. In order to allow occupants of the building to escape, the first consideration is to stop the spread of fire and smoke to other parts of the building, as well as to maintain common escape routes free of fire and smoke. Compartmentation allows the control of fire within a limited space allowing occupants of the building to escape and to control the fire.

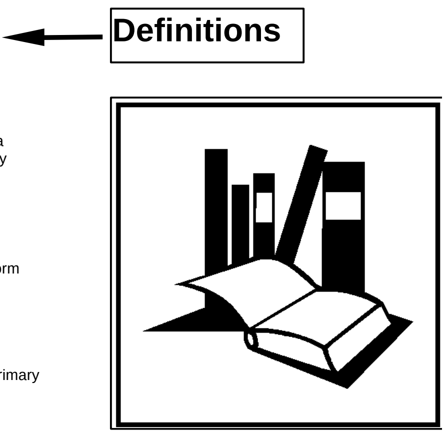
Sub Task Descriptions.

On receipt of the Architects drawings primary and secondary use classes are identified for the whole project. In doing so the shape and size of each use class compartment will be defined. Use class compartments or proposed compartments may need to be sub divided into occupancy types for different owners or tenants. If necessary they may need to be further sub divided to meet maximum permitted floor areaso volumes for given compartments. The final result is the fire compartment.

2: Review and Validate Definitions

Are these definitions correct? Remember that the semantic definition of terms is critical in the definition of an object model for information sharing. Everyone must know the agreed meaning of terms.

Space function:	The occupancy of any space dictated by the role of that space.
Use Class Compartment:	A Compartment of single (Principal) Use with or without Secondary
Primary Use Class:	Primary Use Class as by the Client Brief, ie a primary use of a space ot of spaces.
Secondary Use Class:	The Secondary Use Class is a function ancilary to the primary class.
Proposed Compartment:	Can be: a) a compartment as defined Use Class with related primary and secondary use spaces grouped together to form a single compartment, b) an isolated secondary class, or, c) a group of secondary use classes joined together although related to different primary classes.
Modified Compartment	A proposed compartment divided by occupancy ie ownership or tenancy, to form Mofified
Fire Compartment:	Either a Proposed or Modified Compartment which meets the area volume limits set by requirements or whose area or volume adjusted to meet



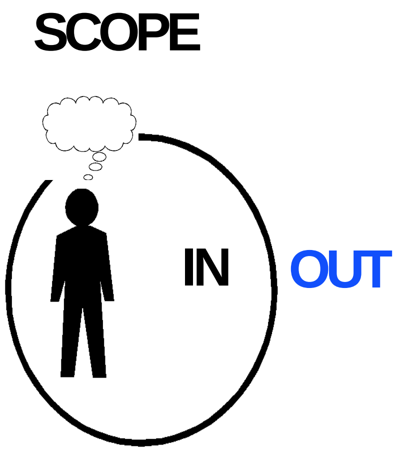
3: Review and Validate Scope

Gain consensus on the proposed scope of the work; not only what is in scope, but also what needs to be stated as being out of scope.

Process Scope: Sub-processes which are within the scope of this process.

Start
Identify primary occupancies.
Identify secondary occupancies.
Identify Use class compartmentation.
Identify Fire Compartmentation.
Finish

Out-of-Scope:
Fire Protection to Space enclosure.
Fire Protection to Elements of structure.
Fire Protection to Electrical, Mechanical and Plumbing Services.
Fire Fighting Equipment
Fire Resistance and Surface Spread of Flame
Interrelationship with adjoining buildings and the boundary.

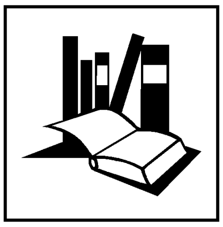


4: Check references

This identifies where basic information is derived from. Are there other references which the authors of the specification should consult?

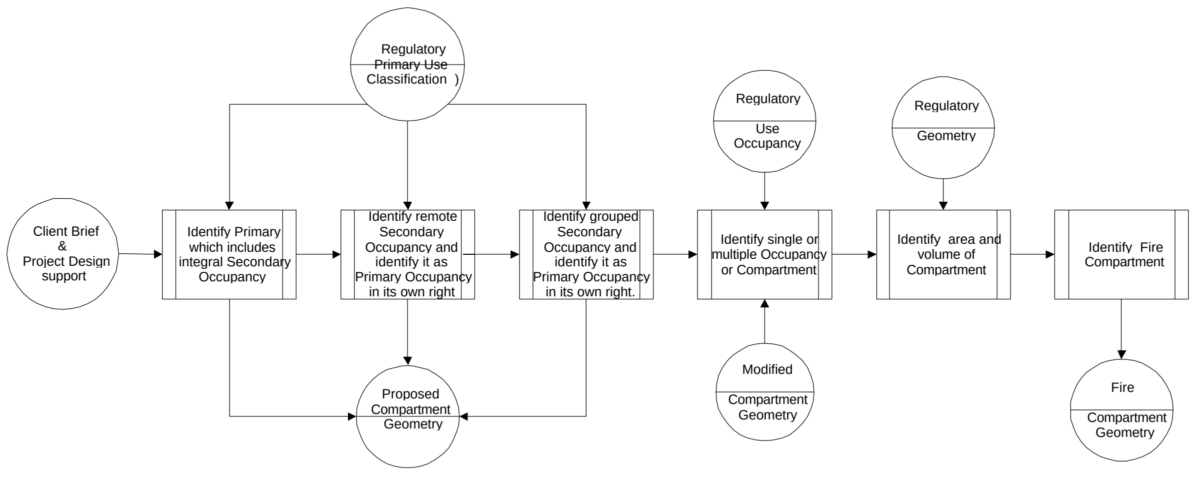
References:
Uniclass classification.
Ci/SfB classification - space classification.

References



5: Review and Validate Overall Process Model

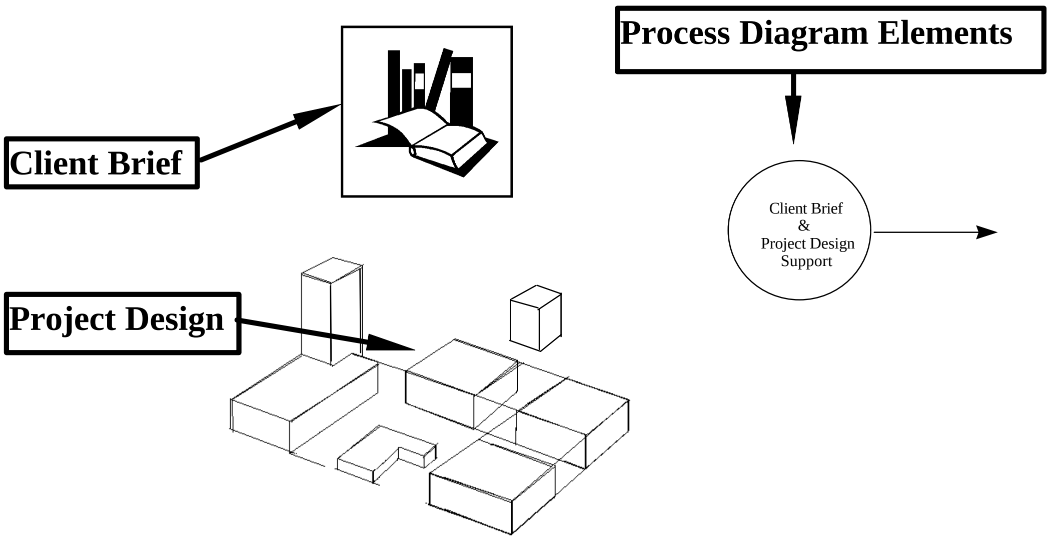
Gaining agreement on the content of the process model is crucial. Having a process which is validated by a substantial peer group means that it is 'probably' right.


Regulatory
Primary Use
Classification)
Regulatory
Use
Occupancy
Regulatory
Geometry
Client Brief
&
Project Design
support
Identify Primary which includes integral Secondary Occupancy
Identify remote Secondary Occupancy and identify it as Primary Occupancy in its own right
Identify grouped Secondary Occupancy and identify it as Primary Occupancy in its own right.
Identify single or multiple Occupancy or Compartment
Identify area and volume of Compartment
Identify Fire Compartment
Proposed
Compartment
Geometry
Modified
Compartment
Geometry
Fire
Compartment
Geometry

Process Analysis

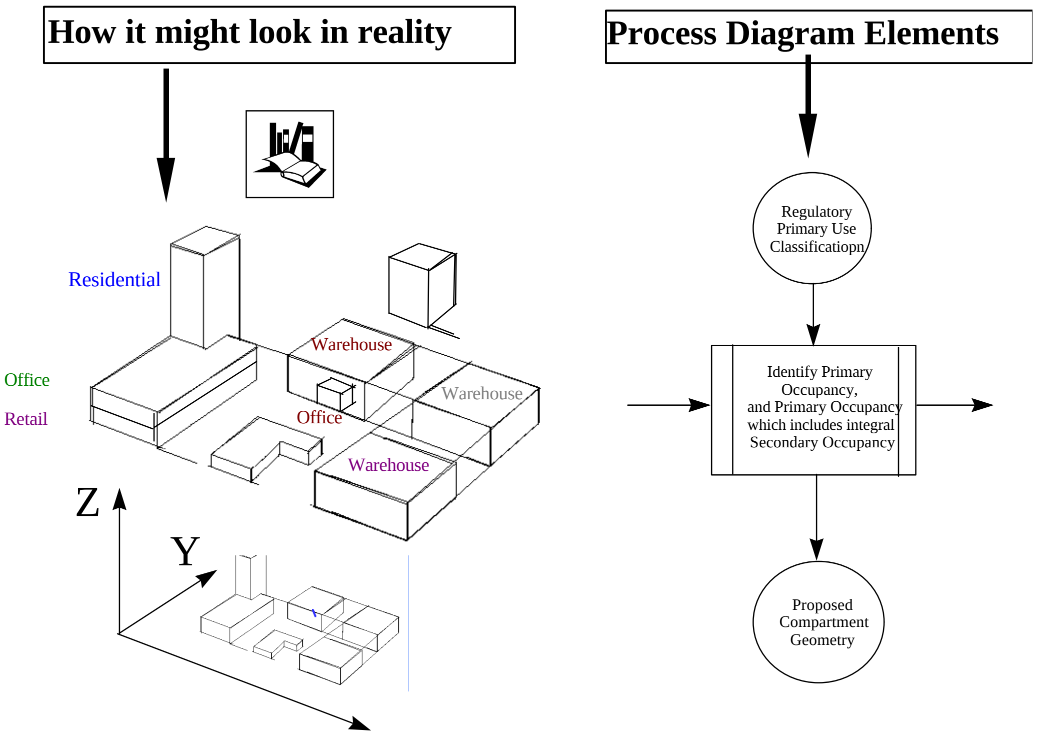
6: Review and Validate Process by Element

Part of the validation includes exhaustively examining the process model on a task by task basis…



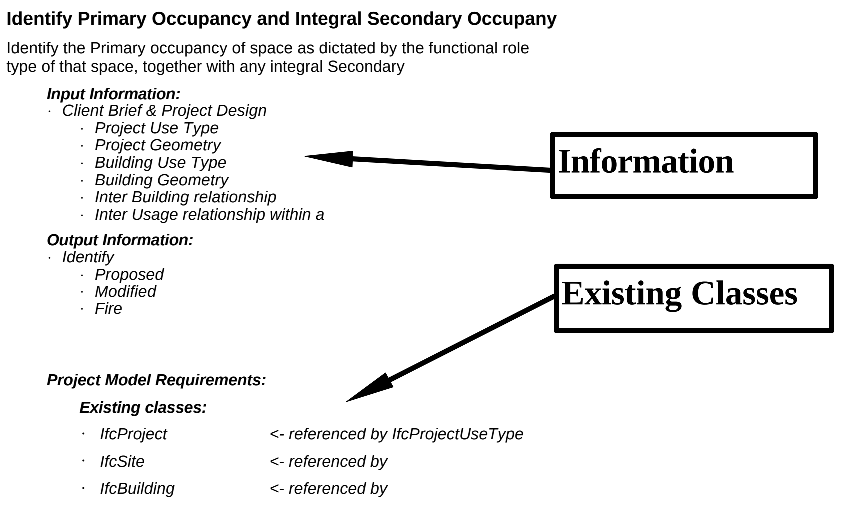
7: Use Diagrams to Make it 'Real' for Reviewers

… which is definitely helped if the reviewers see information in a form which they understand. In general, industry practitioners should not be expected to spend too much time reviewing concepts expressed in formation information language forms like EXPRESS-G or EXPRESS.


How it might look in reality
Process Diagram Elements
Residential
Office
Retail
Warehouse
Office
Warehouse
Warehouse
Z
Y
Regulatory Primary Use Classificatiopn
Identify Primary Occupancy, and Primary Occupancy which includes integral Secondary Occupancy
Proposed Compartment Geometry

8: Review and Validate Information Requirements+Use of Existing Classes

However, it is strongly recommended that specification developers and reviewers look at previous IFC Releases to see what content, if any, they can use directly (and thus save work).


Information
Existing Classes

Identify Primary Occupancy and Integral Secondary Occupany

Identify the Primary occupancy of space as dictated by the functional role type of that space, together with any integral Secondary

Input Information:
- *Client Brief & Project Design*
    - *Project Use Type*
    - *Project Geometry*
    - *Building Use Type*
    - *Building Geometry*
    - *Inter Building relationship*
    - *Inter Usage relationship within a*

Output Information:
- *Identify*
    - *Proposed*
    - *Modified*
    - *Fire*

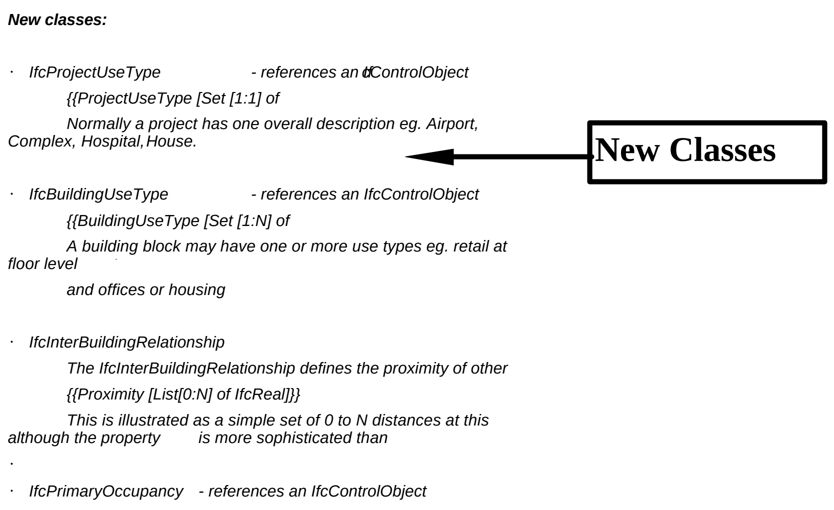
Project Model Requirements:

Existing classes:

· *IfcProject*	*<- referenced by IfcProjectUseType*
· *IfcSite*	*<- referenced by*
· *IfcBuilding*	*<- referenced by*

9: Review and Validate Suggestions for New Classes

But they should also look to see what is missing from previous releases in terms of the proposed process and suggest classes which should be incorporated to support the process requirements.



New classes:

- *IfcProjectUseType - references an IfcControlObject*

  {{ProjectUseType [Set [1:1] of

  Normally a project has one overall description eg. Airport, Complex, Hospital, House.

- *IfcBuildingUseType - references an IfcControlObject*

  {{BuildingUseType [Set [1:N] of

  A building block may have one or more use types eg. retail at floor level

  and offices or housing

- *IfcInterBuildingRelationship*

  The IfcInterBuildingRelationship defines the proximity of other

  {{Proximity [List[0:N] of IfcReal]}}

  This is illustrated as a simple set of 0 to N distances at this although the property is more sophisticated than

- *IfcPrimaryOccupancy - references an IfcControlObject*

Appendix H - Definition of Terms

aggregation	The grouping of items into a construction.
array	A collection of entities in which duplication is allowed and which is indexed by order.
attribute	A property of a class or object.
bag	A collection of entities in which duplication is allowed and in which order is not significant.
boolean	A value which can take on the integer identity 0 or 1.
cardinality	The numerical value that constrains a relation between classes.
class	A description of groups of items or ideas which exhibit common properties and behaviors.
decomposition	The breaking down of a whole into parts.
entity	A unit or ‘thing’ of interest.
instance	An occurrence of a class (see also ‘object’)
integer	A value which has a whole number component only e.g. 1, 22, 6348 etc.
interface	A method of obtaining access to a class or an object functions.
list	A collection of entities in which no duplication is allowed and in which order is significant.
model	A formal statement of classes, properties and behaviors which can be used to inform software implementation and set out requirements for structuring of information exchange and sharing.
object	An item having state, behavior and unique identity.
object model	A representation of information and behaviour in the real world to some acceptable level of detail.
process model	A representation of processes which occur in the real world to some acceptable level of detail.
program	A sequence of executable instructions to a computer
programme	A schedule of actions
real	A value which has a whole number component and a decimal component e.g.
relation	A fact which exists between classes
select	A means of navigating an object model by choosing a class from an available range.
set	A collection of entities in which no duplication is allowed but in which order is not significant.
specification	Domain information for incorporation into a model
string	A value which is alphanumeric e.g. ABC, abc, abc123 etc.
subclass	A class which inherits property and behaviors from a parent or higher level class (superclass)
subtype	see subtype
superclass	A higher level or parent class having one or more subclasses.
supertype	see supertype

Note

- The terms class and entity are used interchangeably in this Guide
- The terms object, instance (when related to a class) and occurrence (when related to a class) are used interchangeably in this Guide.

Appendix I - Abbreviations

COM	Common Object Model (Microsoft)
CORBA	Common Object Request Brokerage Architecture (OMG)
EXCOM	Executive Committee of the International Council of the IAI
FM	Facilities Management
HVAC	Heating, Ventilating and Air Conditioning
IAI	International Alliance for Interoperability
IDEF	Integrated Definition for Function Modeling
IDL	Interface Definition Language
IFC	Industry Foundation Class
IFC-PDEF	IFC Process Definition
ISO	International Standards Organization
ITM	International Technical Management Committee of the IAI
MIDL	Microsoft Interface Definition Language
OMG	Object Management Group
RAC	Research Advisory Committee of the IAI
SIC	Software Implementation Committee of the IAI
STEP	Standard for the Exchange of Product Model Data
STF	Specification Task Force of the IAI
TQM	Total Quality Management

Appendix J - References

Parts referenced from ISO10303 (STEP)

General Parts, Languages and Bindings

- Part 11: EXPRESS Language Reference Manual.
- Part 21: Clear Text Encoding of the Exchange Structure
- Part 22: Standard Data Access Interface Specification
- Part 23: SDAI C++ Binding.

Conformance Testing

- Part 31. General Concepts.
- Part 33: Structure and Use of Abstract Test Suites.

Integrated Generic and Application Resources

- Part 41: Fundamentals of Product Description.
- Part 42: Geometric and Topological Representation.
- Part 43: Representation
- Part 106WD. Building Construction Core Model.

Application Protocols

- Part 225: Building Elements Using Explicit Shape Representation.

Parts referenced from OMG CORBA

- Interface Definition Language Specification